THE UNIVERSITY OF NEW MEXICO

ECE 538
Advanced Computer Architecture

Instructor: Lei Yang

Department of Electrical and Computer Engineering

November 2, 2021

2021-11-01: End Class 20 here

- ✓ Pipeline and Pipeline Hazards
  - Structural Hazards
  - Data Hazards
  - Control Hazards

Limitations of ILP for parallelable Processors

- Clock rate, logic complexity, and power
- Small issue rate and load/store buffers
- Longer latency

What Is Ahead??

What Is Ahead??

- ~2000, Instruction-Level Parallelism (ILP) in its peak
- ~2005, Multicore processor for better performance
- Superscalar processor & Large number of cores
- Data-Level Parallelism (DLP)
- Thread-Level Parallelism (TLP)

→ SIMD and Variations (Vector, SIMD ISE, GPUs)

Data Level Parallelism
(DLP)

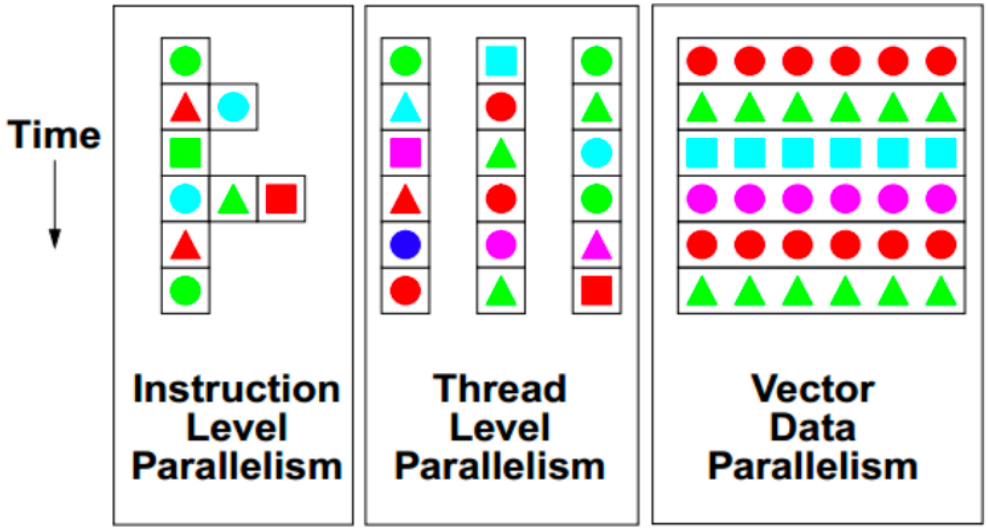
[Ref] Chapter Four in the textbook

Overview

- Vector Architecture
- Multimedia SIMD instruction set extensions
- Graphics processing units (GPUs)

Overview – Three Levels of Parallelism

- ILP: Instruction Level Parallelism – e.g., Superscalar and VLIW processor
  - Out-of-Order execution (all in hardware)
  - Multiple independent instructions are identified and grouped to be executed concurrently in different functional units in a single processor
  - Can reduce CPI to values less than 1



Overview – Three Levels of Parallelism

- Other forms of parallelism
  - TLP: Thread/Task Level Parallelism – e.g., multicore and multi-processor systems
    - Multiprocessors, and HW Multithreading
    - Several independent threads/tasks are executed simultaneously
    - Reduce total execution time of multiple tasks
  - DLP: Data Level Parallelism – e.g., Vector processor and array processor
    - ✓ The same operation is performed on multiple data values concurrently in multiple processing units
    - ✓ Reduce instruction count to enhance performance

Review

- ILP: Instruction Level Parallelism
  - Out-of-Order execution (all in hardware)
  - Pipeline: overlap processing stages of different instructions
  - SISD, IPC hardly achieves more than 2

Fundamental Problems of ILP

Clock rate and IPC are at odds with each other:

- Pipelining: Fast clock; Increased hazards lower IPC
- Wide issue: Higher IPC; Slow down clocks

- Other forms of parallelism
  - DLP: Data Level Parallelism
    - Vector Processors, SIMD extensions, and GPUs
  - TLP: Thread Level Parallelism
    - Multiprocessors, and HW Multithread

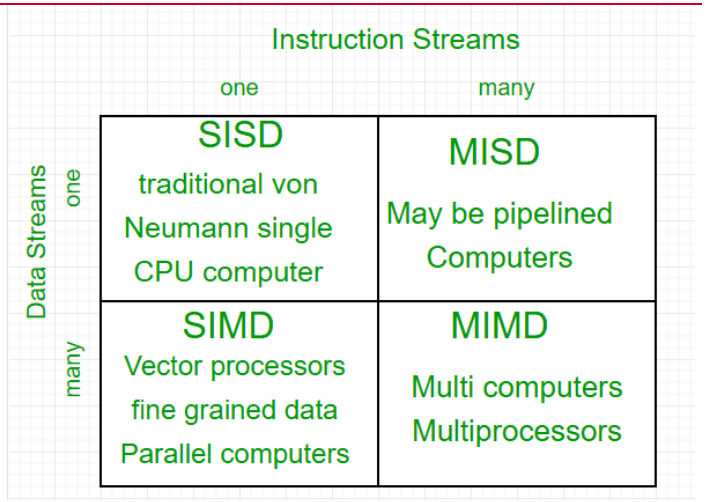
Why DLP?

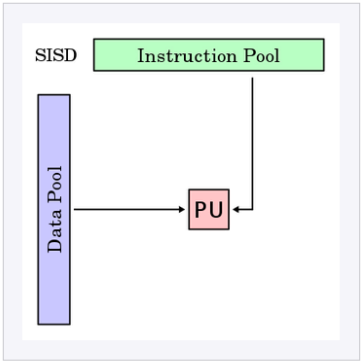
- Single operation repeated on multiple data elements
- Less general than ILP: parallel instructions are same operations

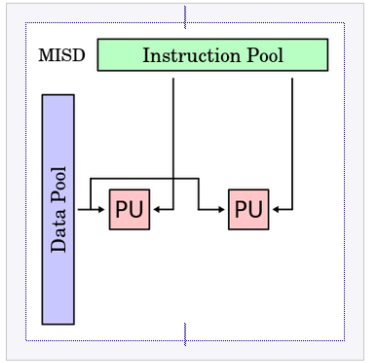
DATA LEVEL PARALLELISM (DLP)

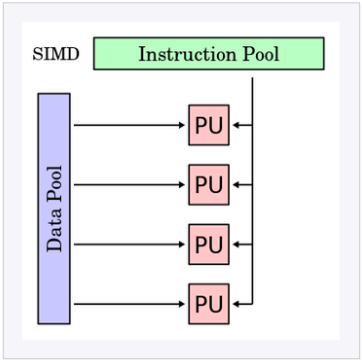
- Computing Models-Flynn's Classification

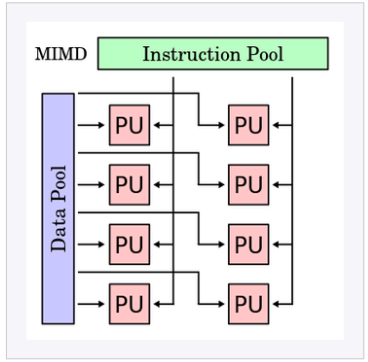
- **SISD** -- Single Instruction Single Data stream
  - Conventional sequential processor
  - Not suitable to realize data-level parallelism
- **SIMD** -- Single Instruction Multiple Data stream
  - Efficiently utilize the data-level parallelism
- **MISD** -- Multiple Instruction Single Data stream
  - No commercial programmable systems
- **MIMD** -- Multiple Instruction Multiple Data stream
  - True multiprocessor system that issues multiple instructions
  - Can support data-level parallelism







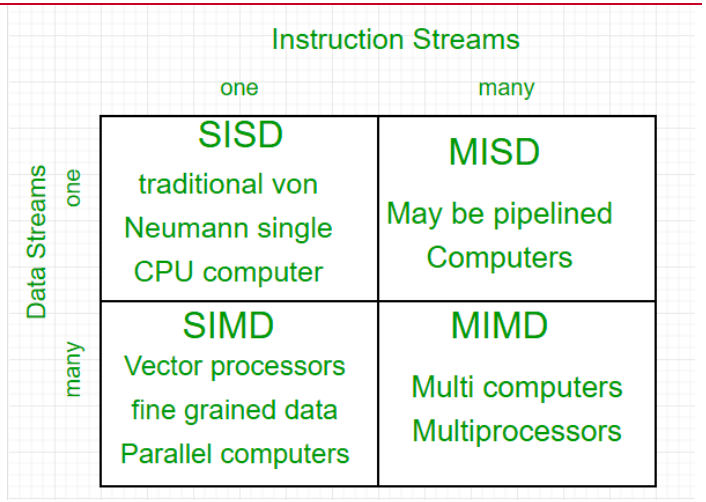


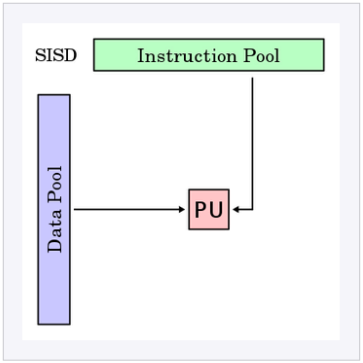


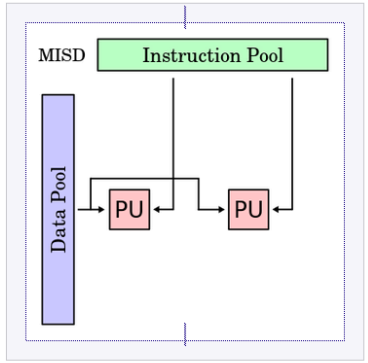
DATA LEVEL PARALLELISM (DLP)

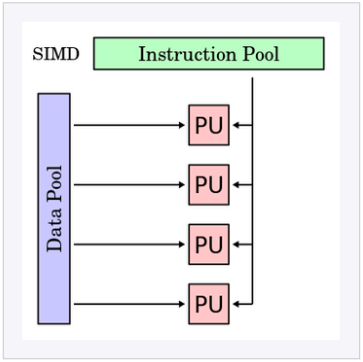
- Computing Models-Flynn's Classification

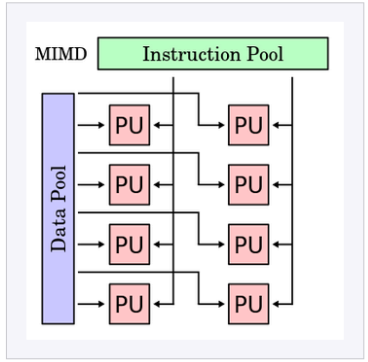
- **SISD** -- Single Instruction Single Data stream
  - Conventional sequential processor
  - Not suitable to realize data-level parallelism
- **SIMD** -- Single Instruction Multiple Data stream
  - Efficiently utilize the data-level parallelism
- **MISD** -- Multiple Instruction Single Data stream
  - No commercial programmable systems
- **MIMD** -- Multiple Instruction Multiple Data stream
  - True multiprocessor system that issues multiple instructions
  - Can support data-level parallelism





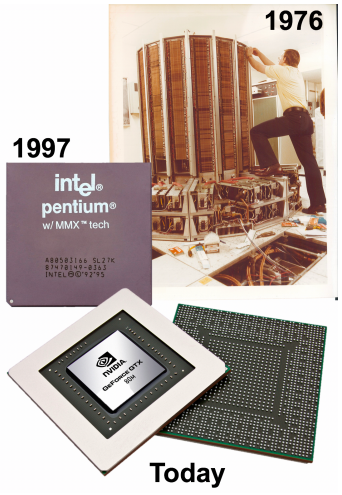






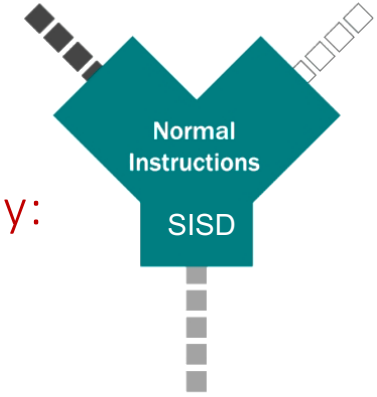
Data Level Parallelism (DLP)

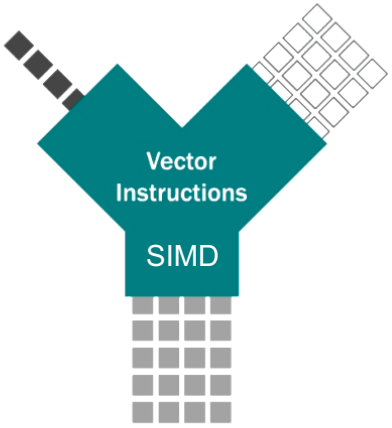
- DLP:
  - Common in scientific computing
  - Executing the same code on a large number of objects



- DLP architectures - variations of SIMD
  - **Vector processors** – e.g., Cray Machines
  - **SIMD Extensions** – e.g., Intel MMX
  - **Graphics Processing Unit** – e.g., NVIDIA

- Improving throughput rather than latency:
  - Not good for non-parallel workloads



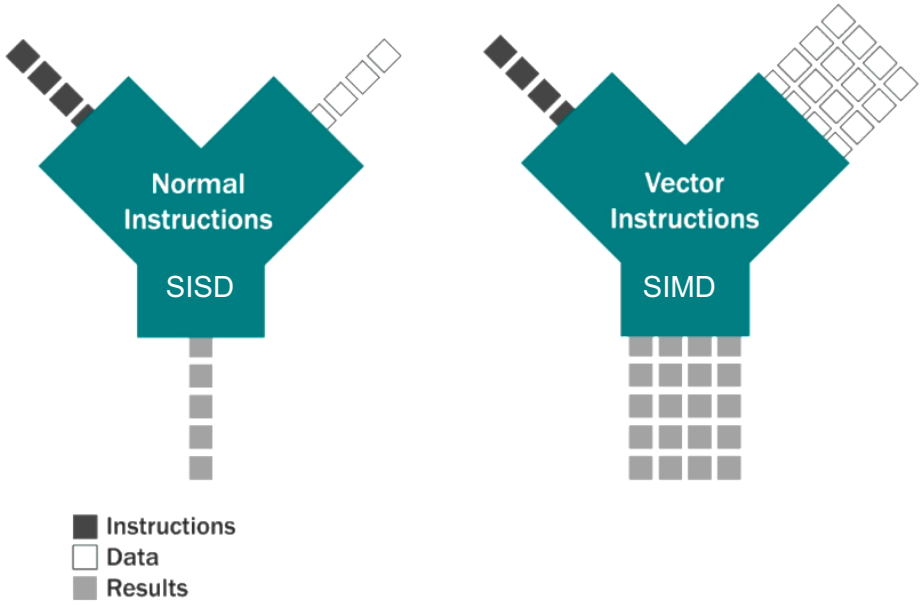


DLP AND SIMD

- The Idea:
  - Reduce instruction processing overhead and leverage the fact that certain applications need to do the same operation on large amounts of data
- Examples:
  - Graphics Processing Unit(GPU), Audio encoding/decoding, physics simulations, Deep Neural Networks (DNN), etc.

```
for (I = 0; I < 100; I++)
    Z[I] = A*X[I] + Y[I];

L0: ldf X(r1),f1      // I is in r1
    mulf f0,f1,f2     // A is in f0
    ldf Y(r1),f3
    addf f2,f3,f4
    stf f4,Z(r1)
    addi r1,4,r1
    blti r1,400,L0
```



Example of DLP: inner loop-level parallelism

✓ Iterations can be performed in parallel

DLP AND SIMD

- SIMD Machines:
  - SIMD instructions set extensions (ISEs) for traditional processors
  - Vector processors, which are really just dedicated SIMD processors
  - Graphic Processor Units (GPUs), which have large number of SIMD processors

- SIMD Instruction Set Extensions (ISEs):
  - Primarily been included into Intel and ARM processors for multimedia
  - Have far reaching applications, e.g., *cryptography*
  - Intel SIMD ISEs

    -- MultiMedia eXtensions MMX, Streaming SIMD, Extensions (SSE), Advanced Vector Extensions, (AVX), AVX2, and AVX-512

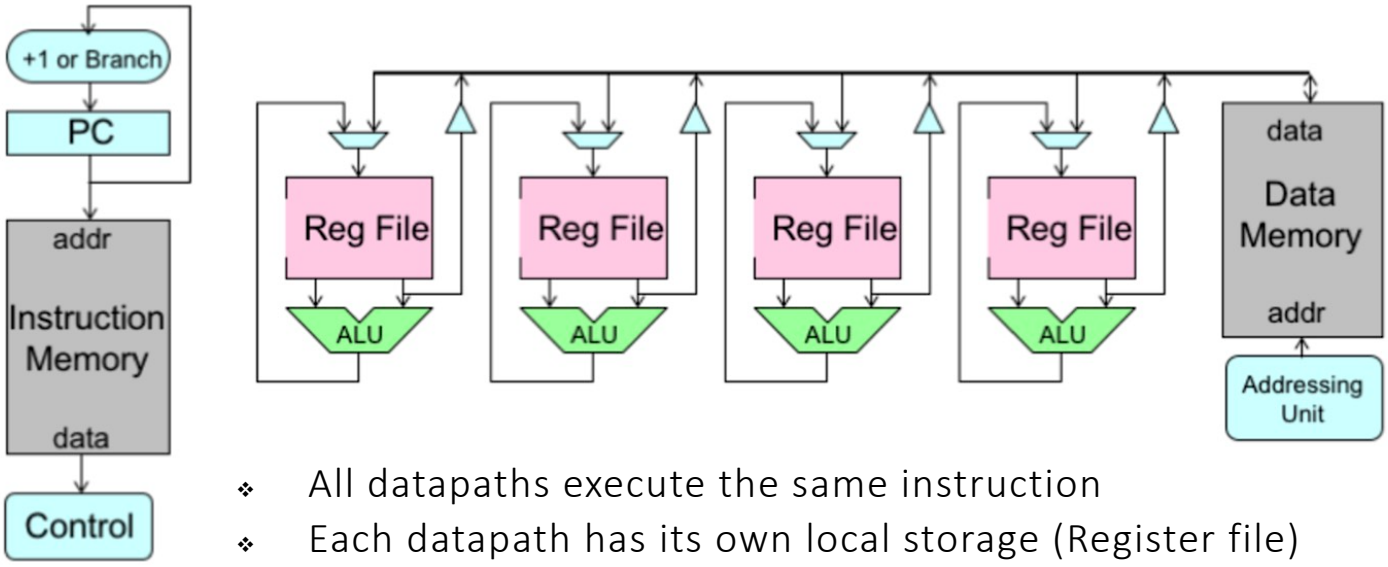
    -- Gradually turned the 07 into a vector processor

DATA LEVEL PARALLELISM (DLP) - VECTORS

- One way to exploit DLP: Vectors
  - Some operations applied to multiple data elements

```
for (int i = 0; i < 16; i ++)   x[i] = a[i] + b[i];
```

  - Extend processor with vector "Data Type" (vector processors) or vector ISA extensions



- Vector: Array of MVL 32-bit FP numbers
  Maximum vector length (MVL): 8-64
- Vector register file: 8-16 vector registers

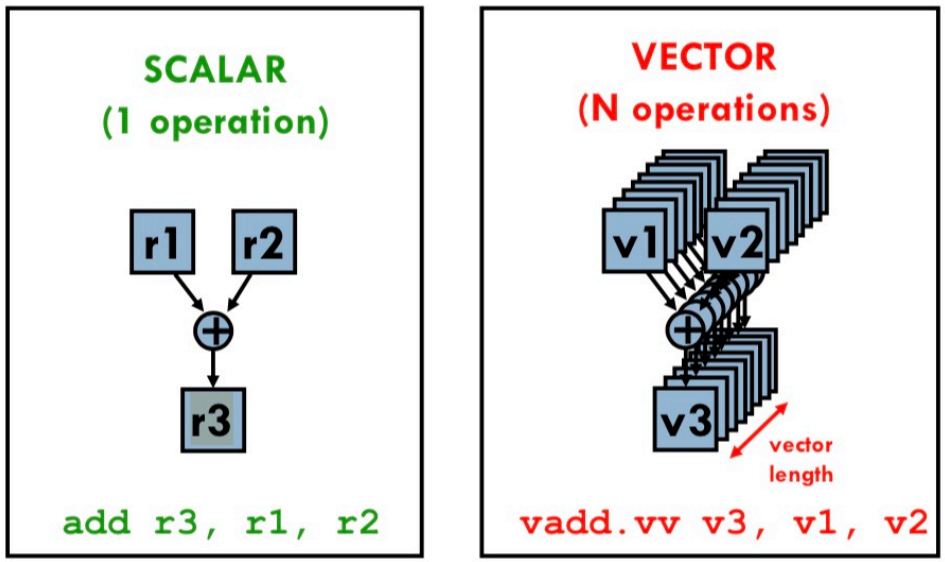
  - All datapaths execute the same instruction
  - Each datapath has its own local storage (Register file)
  - Memory access with vector loads and stores + wide memory port

DATA LEVEL PARALLELISM (DLP) - VECTORS

- Why Vectors?
  - Vector instructions allow deeper pipelines
    - ✓ No intra-vector interlocks
    - ✓ No intra-vector hazards
    - ✓ Inner loop control hazards eliminated
    - ✓ Need not issue multiple instructions
    - ✓ Vectors can present memory access pattern to HW
  - Want deeper pipelines but…
    - ✓ Interlock logic is hard to divide into more stages
    - ✓ Bubbles due to data hazard increase
    - ✓ Hard to issue multiple instructions per cycle
    - ✓ Fetch & Issue bottleneck (Flynn Bottleneck)

Data Level Parallelism (DLP) - Vectors

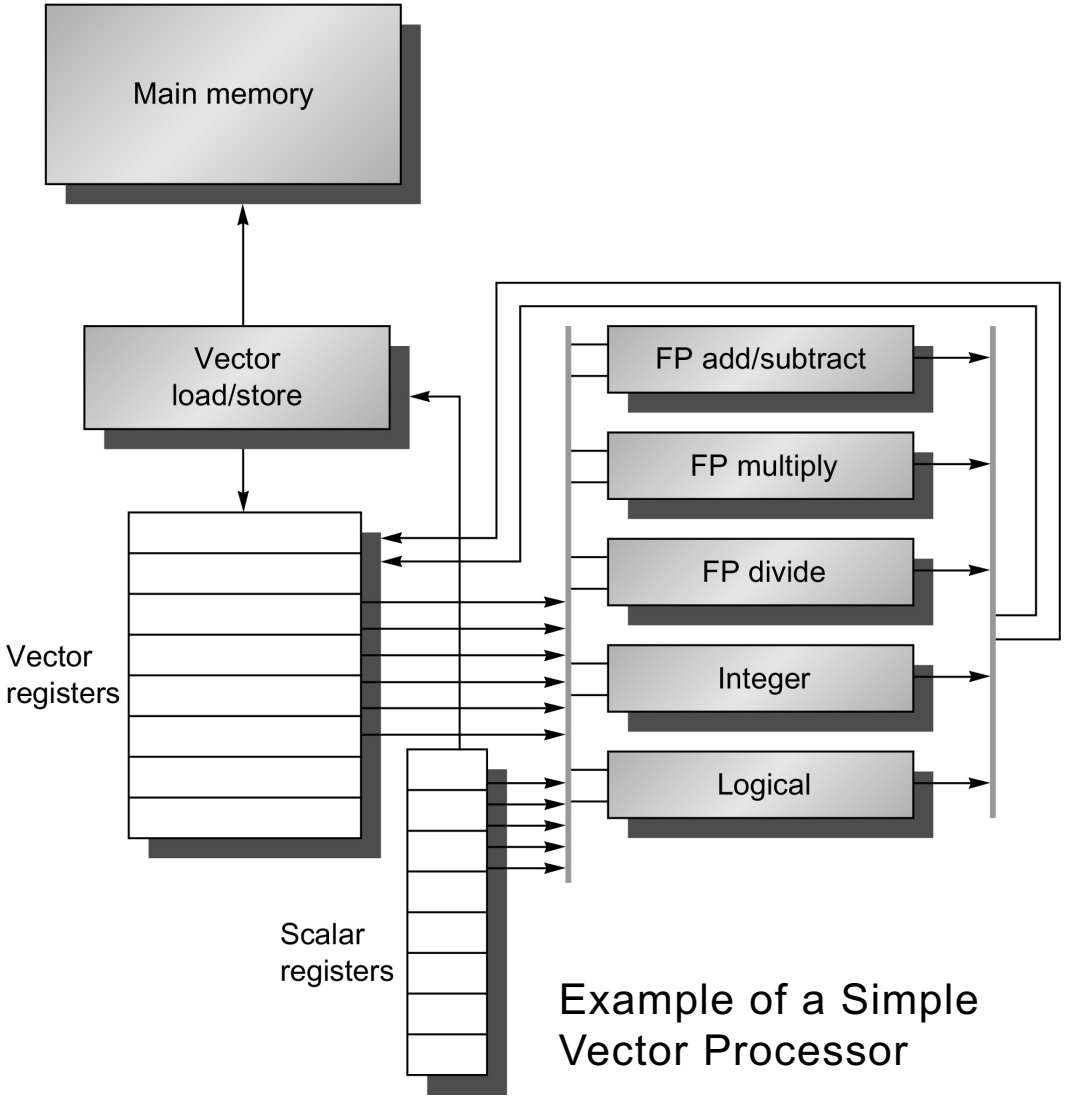
- What is a Vector Processor?
  - Scalar processors operate on single numbers (scalars)
  - Vector processors operate on linear sequences of numbers (vectors)



- 1976,eEarly example was Cray-1 installed at Los Alamos
- RISC-V has created a vector processor extension to ISA (RV64V)
- Intel processors with the AVX2 are pretty much vector processors...

DATA LEVEL PARALLELISM (DLP) - VECTORS

- What is a Vector Processor?
  - Typically has a traditional scalar processor with a large vector register file that supports a varied number of registers and sizes
  - Has multiple functional units that are arranged in lanes
  - Has vector load/store instructions including scatter/gather instructions that gather sparse data from memory into a single vector and then scatter the result back out into memory



Example of a Simple Vector Processor

DATA LEVEL PARALLELISM (DLP) - VECTORS

- What are in a Vector Processor?
  - A scalar processor (e.g., a MIPS processor)
    - ✓ Scalar register file (32 registers)
    - ✓ Scalar functional units (arithmetic, load/store, etc)
  - A vector register file (a 2D register array)
    - ✓ Each register is an array of elements (e.g., 32 registers with 32 64-bit elements per register)
    - ✓ MVL (Maximum Vector Length) = Max # of elements per register
  - A set of vector functional units
    - ✓ Integer, FP, load/store, ect
    - ✓ Sometimes vector and scalar units are combined (share ALUs)

DATA LEVEL PARALLELISM (DLP) - VECTORS

- Vector Length (VL)
  - Basic:
    - ✓ Fixed vector length (typical in narrow SIMD)
    - ✓ Is this efficient for wide SIMD? (e.g., 32-wide vectors)→ NO
  - Vector-length (VL) register:
    - ✓ Control length of any vector operation, including vector loads and stores
    - ✓ VL can be set up to MVL (e.g., 32)

DATA LEVEL PARALLELISM (DLP) - VECTORS

- Vector ISA:
  - Single instruction defines multiple operations
    - ✓ Lower instruction fetch/decode/issue cost
  - Operations are executed in parallel
    - ✓ Naturally no dependency among data elements
    - ✓ Simple hardware
  - Predictable memory access pattern
    - ✓ Improve performance via prefetching
    - ✓ Simple memory scheduling policy
    - ✓ Multi banking may be used for improving bandwidth

DATA LEVEL PARALLELISM (DLP) - VECTORS

- Advantages of Vector ISAs
  - ++ **Compact**: Single instruction defines N operations
    - ✓ Amortize the cost of instruction fetch/decode/issue
    - ✓ Also reduce the frequency of branches
  - ++ **Parallel**: N operations are (data) parallel
    - ✓ No dependencies
    - ✓ No need for complex HW to detect parallelism
    - ✓ Can execute in parallel assuming N parallel data paths
  - ++ **Expressive**: memory operations describe patterns
    - ✓ Continuous or regular memory access pattern
    - ✓ Can prefetch or accelerate using wide/multi-banked memory
    - ✓ Can amortize high latency for 1st element over large sequential pattern

DATA LEVEL PARALLELISM (DLP) - VECTORS

- Disadvantages of Vector ISAs
  - -- Works (only) if parallelism is regular (data/SIMD parallelism)
    - ✓ *Very inefficient if parallelism is irregular*
- Limitations of Vector ISAs
  - -- Memory (bandwidth) can easily become a bottleneck, especially if
    - ✓ Compute/memory operation balance is not maintained
    - ✓ Data is not mapped appropriately to memory banks

Vector Execution Time: the vector execution time depends on three factors

- Length of operand vectors
- Structural hazards
- Data dependences

DATA LEVEL PARALLELISM (DLP) - VECTORS

- Conditional execution

- **Questions**: How to handle branches? → *Prediction*
  - ✓ Use masks, flag vectors with single-bit elements
  - ✓ Determine the flag values based on vector comparison
  - ✓ Use flag registers as control masks for the next vector operations

```
for (i = 0; i < 100; i++){
    if (A(i)!=B[i])
        A[i]-=B[i];
}
```

```
vld V1, Ra
vld V2, Rb
vcmp.neq.vv M0, V1, V2
vsub.vv V3, V2, V1, M0
vst V3, Ra
```

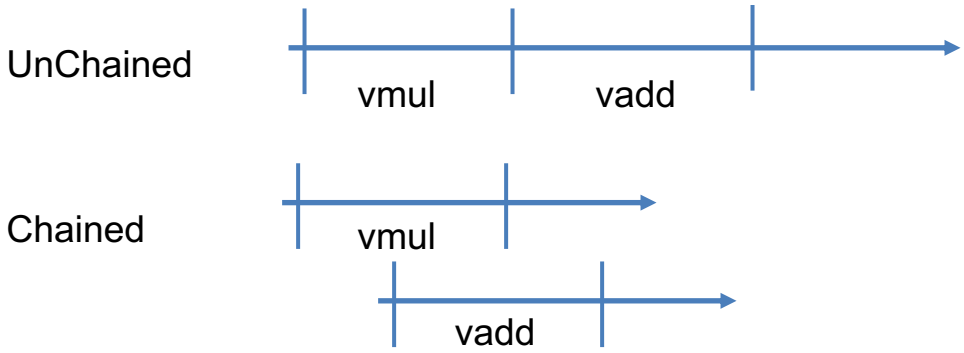
DATA LEVEL PARALLELISM (DLP) - VECTORS

Optimization I: Chaining

- Suppose the following code with VL = 32:

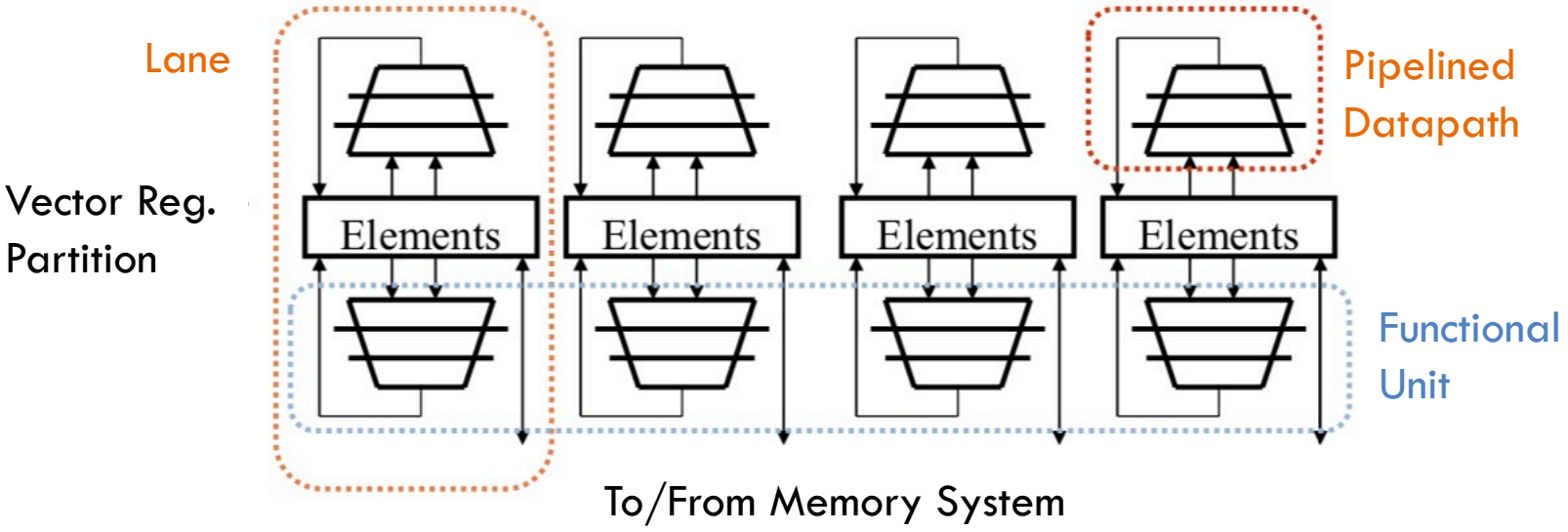
```
vmul.vv V1,V2,V3
vadd.vv V4,V1,V5     # very long RAW hazard
```

- Chaining
  - V1 is not a single entity but a group of individual elements
  - Pipeline forwarding can work on an element basis
- Flexible Chaining:
  - Allow vector to chain to any other active vector operation => more read/write ports



DATA LEVEL PARALLELISM (DLP) - VECTORS

- Optimization II: Multiple Lanes

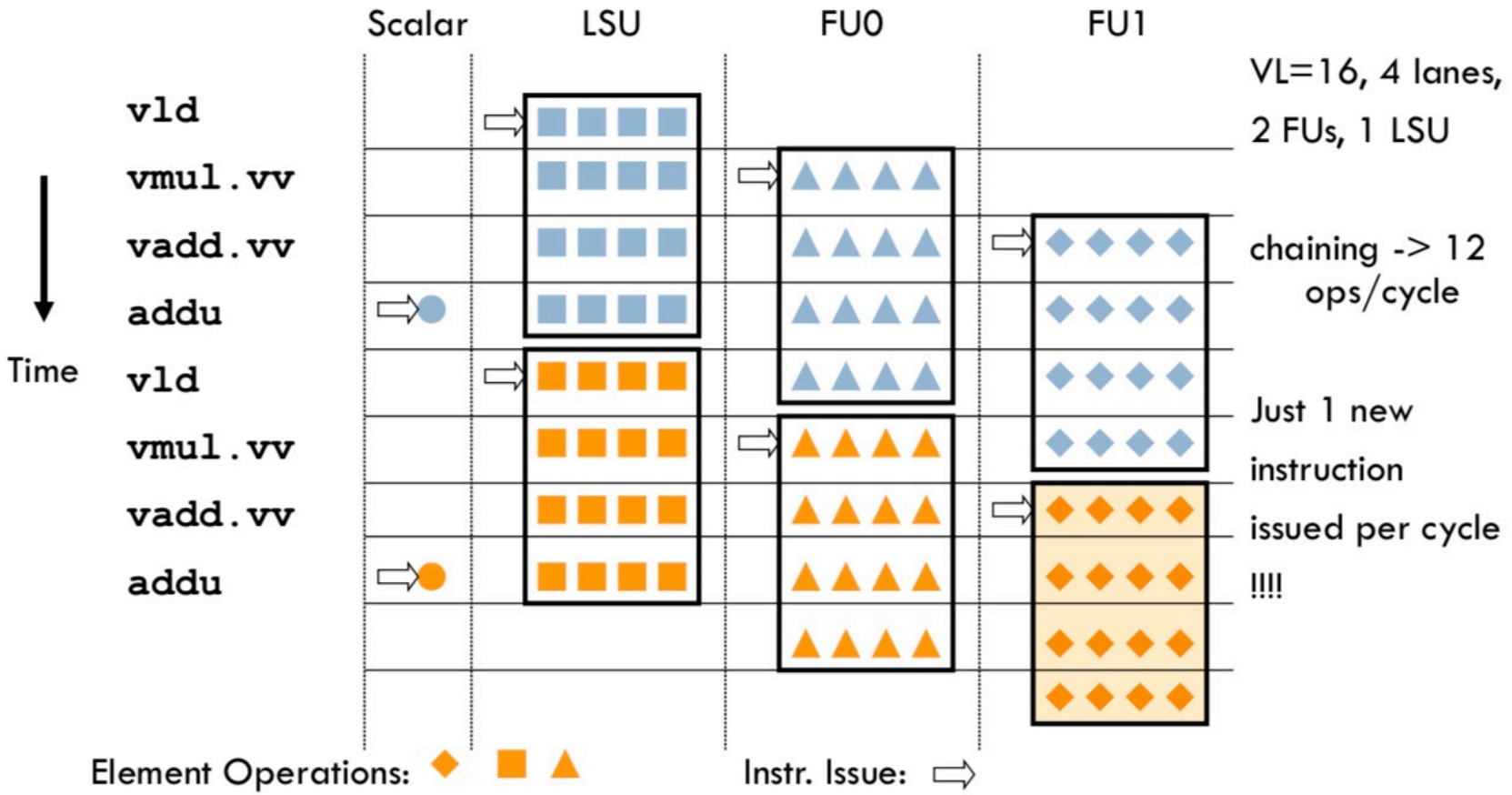


- Modular, scalar design
  - Elements for each vector register are interleaved across the lanes
  - Each lane receives identical control
  - Multiple element operations are executed per cycle
  - No need for inter-lane communication for most vector instructions

DATA LEVEL PARALLELISM (DLP) - VECTORS

- Optimization II: Multiple Lanes

- Example of Chaining & Multi-lane



DATA LEVEL PARALLELISM (DLP) - VECTORS

Optimization III: Conditional Execution

- Suppose you want to vectorize this:

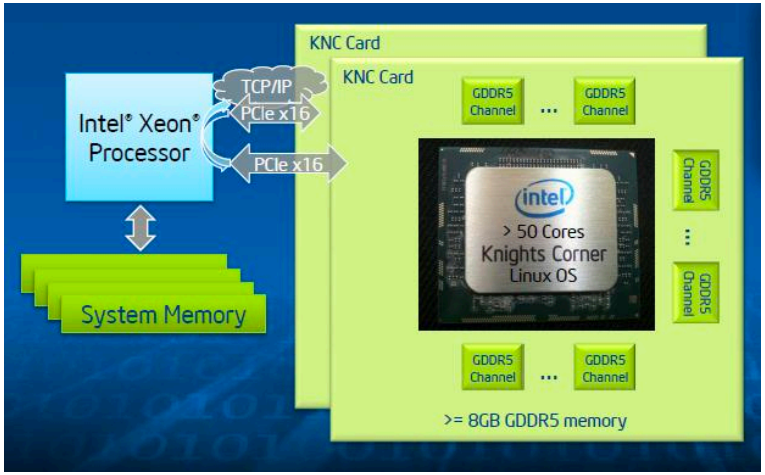
```
for (i = 0; i < 100; i++){
    if (A(i)!=B[i])
        A[i]-=B[i];
}
```

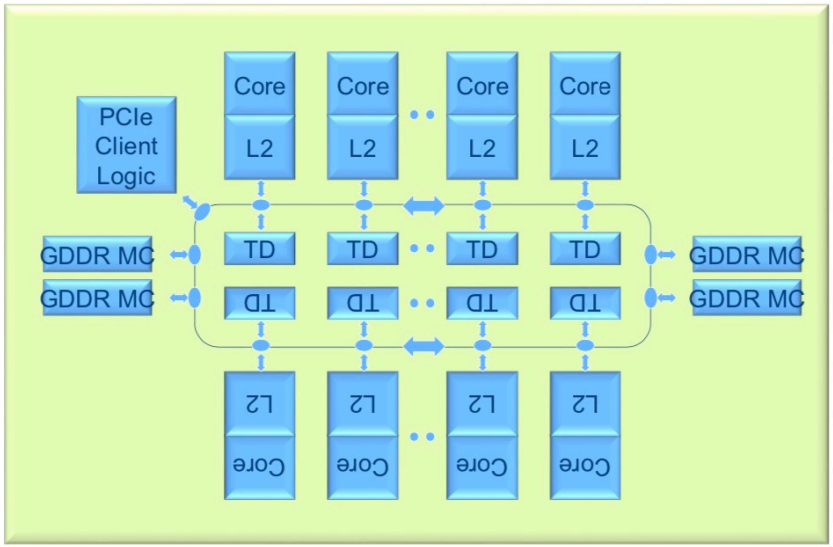
- Solutions: Vector conditional execution (Prediction)
  - **Add vector flag registers with single-bit elements (masks)**
  - **Use a vector compare to set the flag register**
  - **Use flag register as mask control for the vector sub**: add executed only for vector elements with corresponding flag element set
- Vector Code

```
vld V1, Ra
vld V2, Rb
vcmp.neq.vv M0, V1, V2      # vector compare
vsub.vv V3, V2, V1, M0      # conditional vadd
vst V3, Ra
```

DATA LEVEL PARALLELISM (DLP) – SIMD ISE

- SIMD: Intel Xeon Phi (Knights Corner)



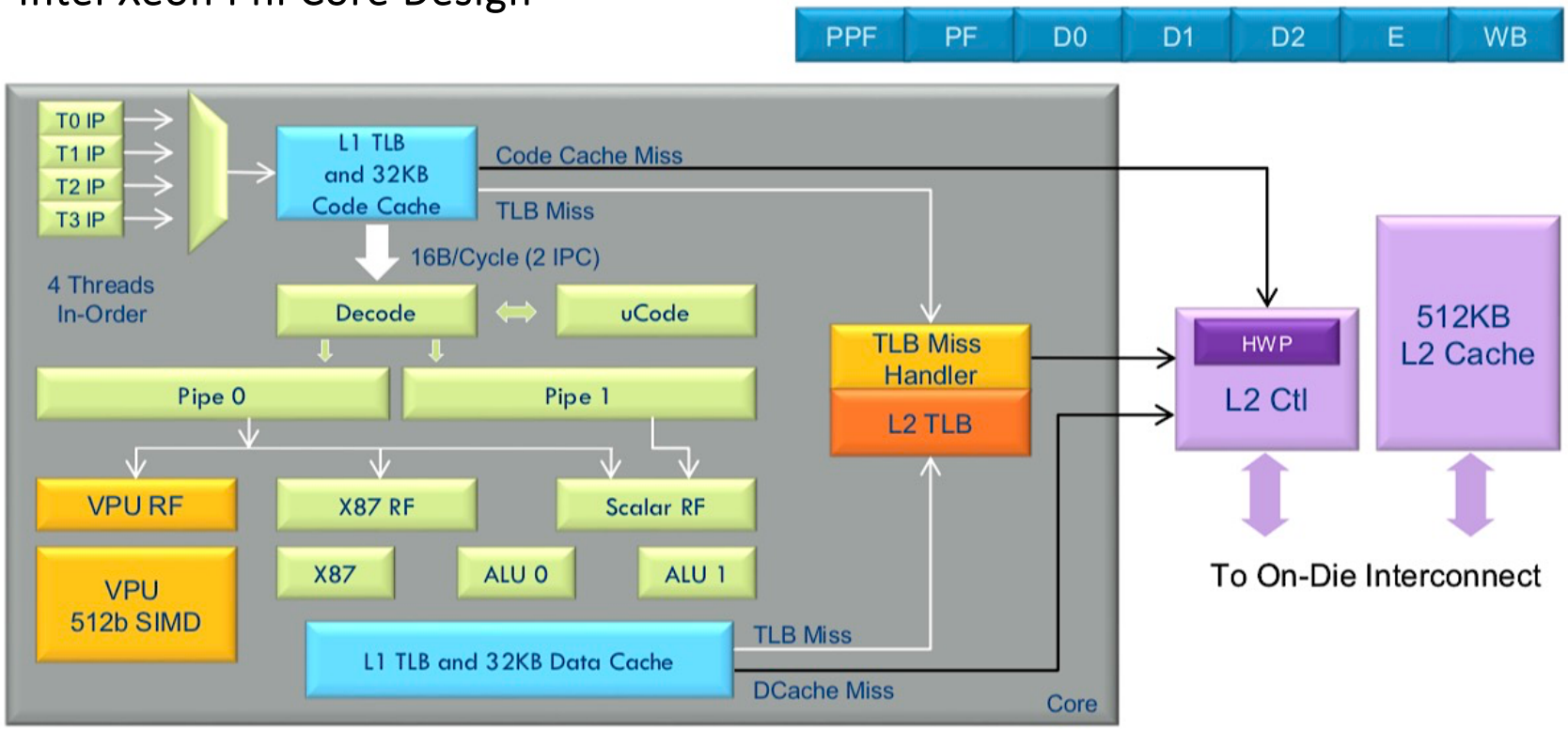


The first-generation Intel® Xeon Phi™ product codenamed “Knights Corner”

- A multi-core chip with x86-based vector processors
  - ✓ Ring interconnect, provide L2 caches, coherent
- Targeting the HPC market
  - ✓ Goal: high GFLOPS, GFLOPS/Watt

DATA LEVEL PARALLELISM (DLP) - SIMD

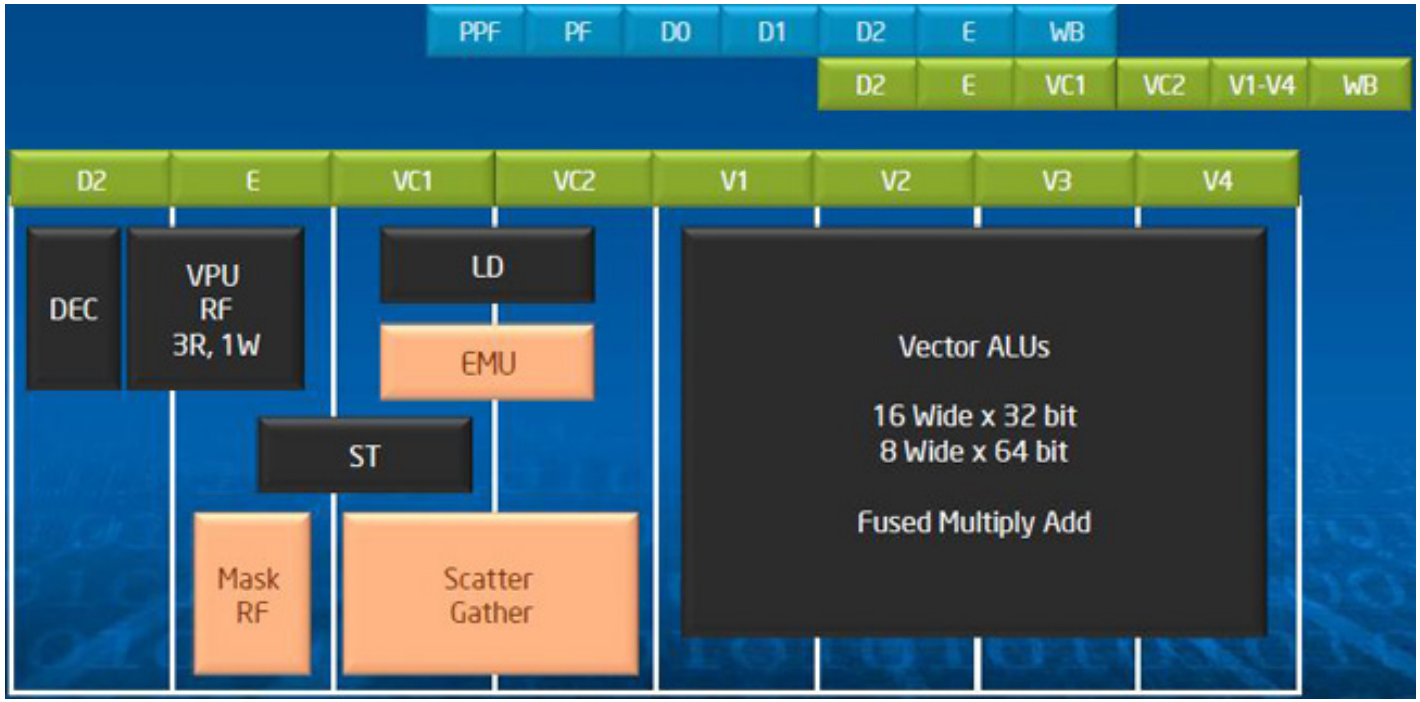
- Intel Xeon Phi Core Design



- ✓ 4-way threaded + vector processing
- ✓ In-order, short-pipeline
- ✓ Vector ISA: 32 vector registers (512b), 8 mask registers, scatter/gather

DATA LEVEL PARALLELISM (DLP) - SIMD

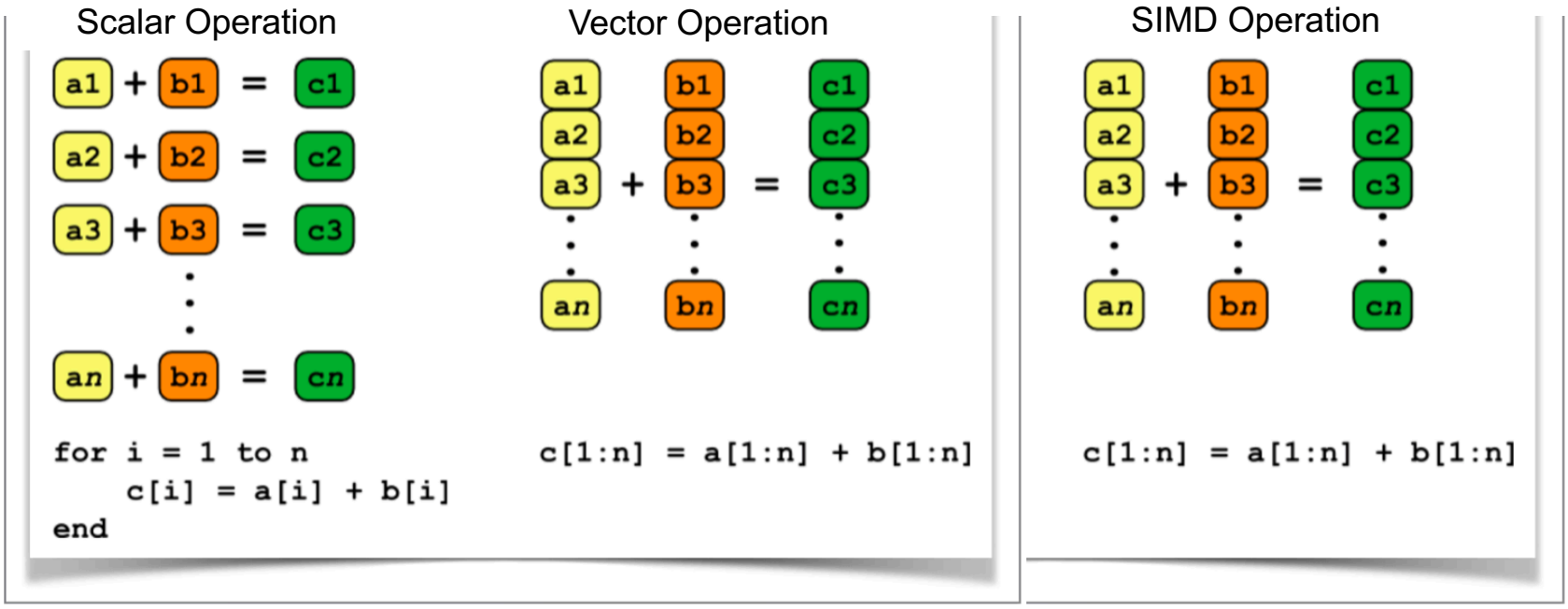
- Intel Xeon Phi Core Design



- ✓ Vector processing unit (VPU): 32 vector registers (512b), 8 mask registers, scatter/gather
- ✓ Targeting the High-Performance Computing market

Data Level Parallelism (DLP)

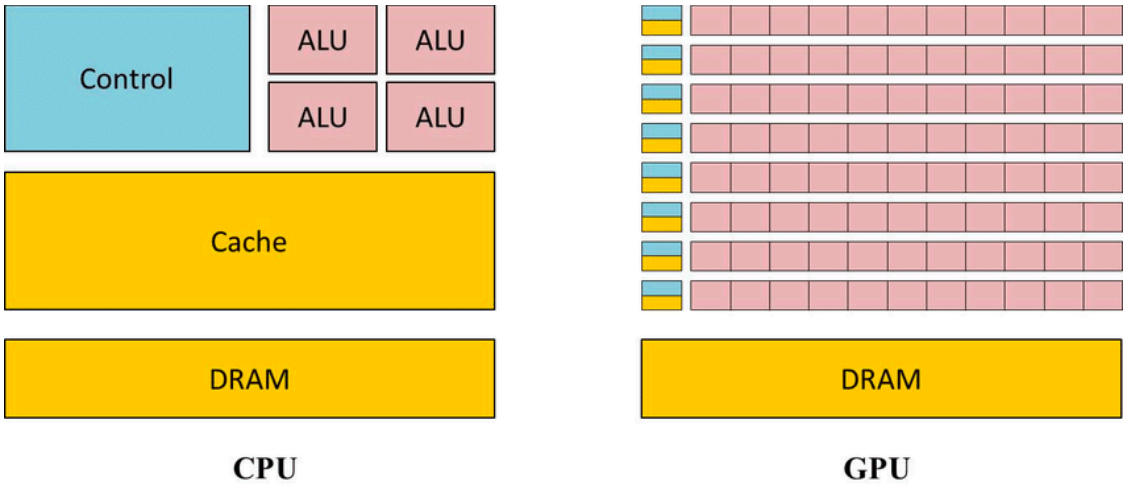
- Vector Processor vs. SIMD
  - Vector processors are typically more flexible with vector length allowing programmers to have larger but few vector registers or smaller but more vector registers
  - SIMDs have rigid vector registers
  - Vector processors have scatter/gather instructions that move data between spare memory and dense registers



DATA LEVEL PARALLELISM (DLP) - GPU

- GPUs – An architecture For compute-intensive, highly data parallel computation

- Have a long history that runs in parallel with vector processors
- Originally designed to offload graphics rendering from CPU
- Fast-growing video game industry exerts strong economic pressure forces constant innovation
- Recently used for a variety of scientific, ML applications, i.e., General Purpose GPU (GPGPU)

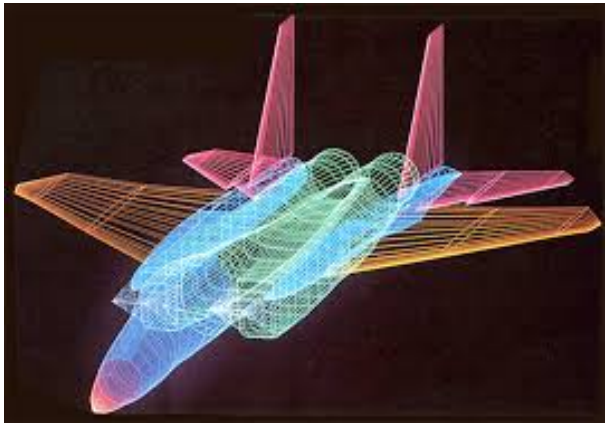


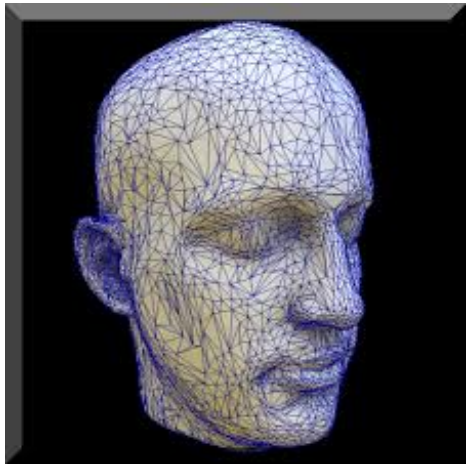
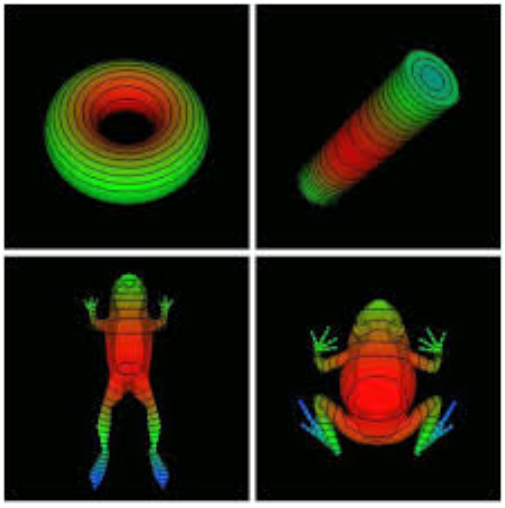
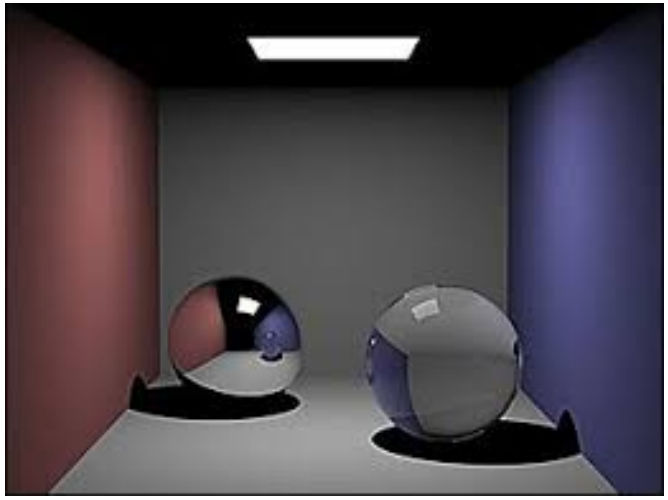
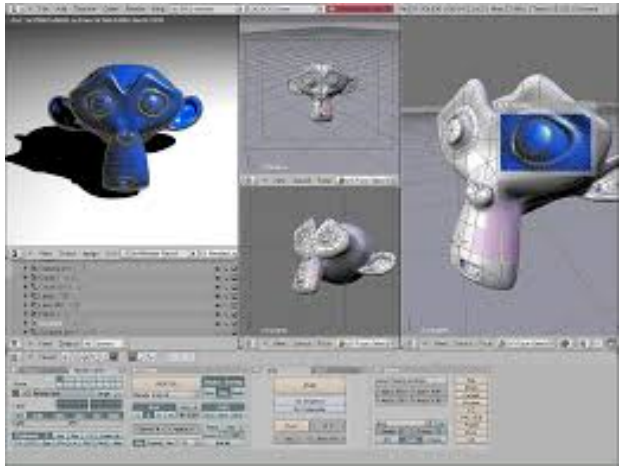
- ✓ Massive SIMD machines that exploit ILP, TLP, and DLP
- ✓ Only recently have L1 caches been included in GPUs
- ✓ Access is typically to GDDR which is a special parallel DRAM
- ✓ Local SRAM is available for SIMD processor as a scratch pad memory (SPM)

Graphics Processing Unit (GPU)

Data Level Parallelism (DLP)

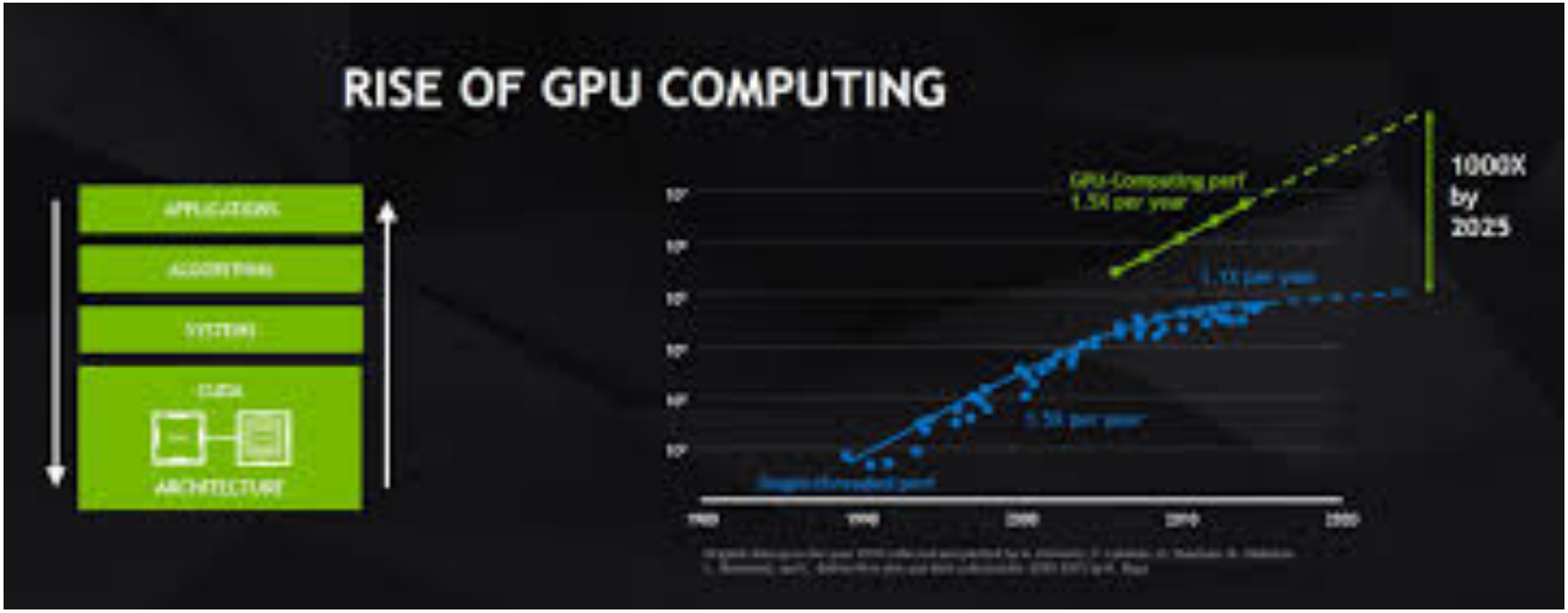
- Computer Graphics



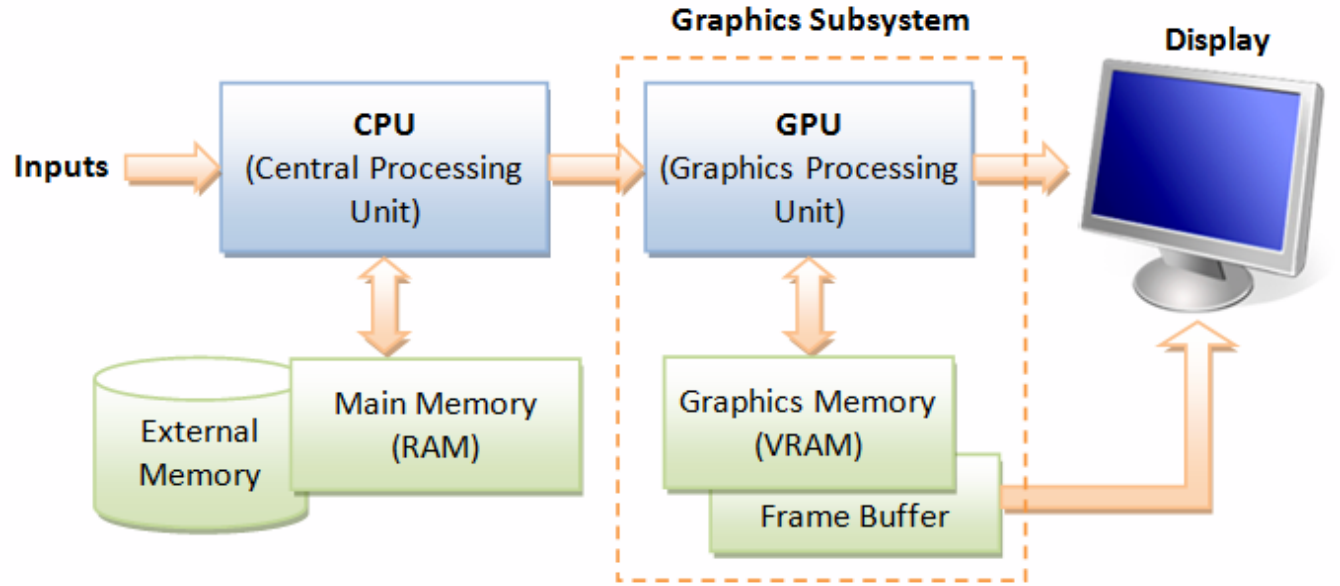





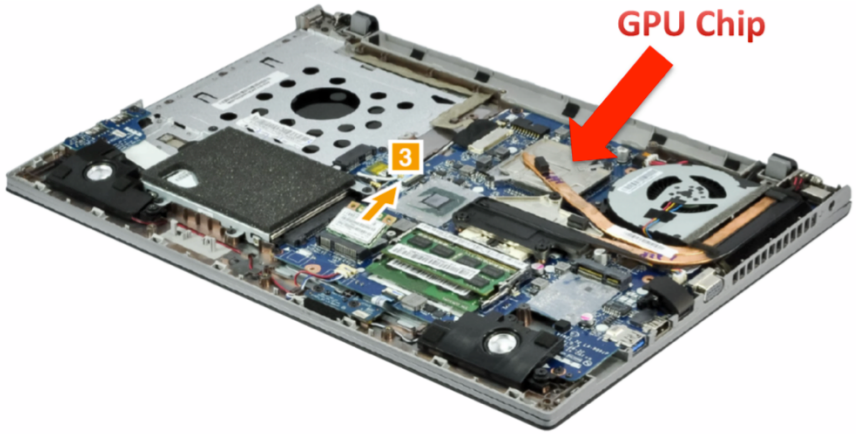
DATA LEVEL PARALLELISM (DLP)

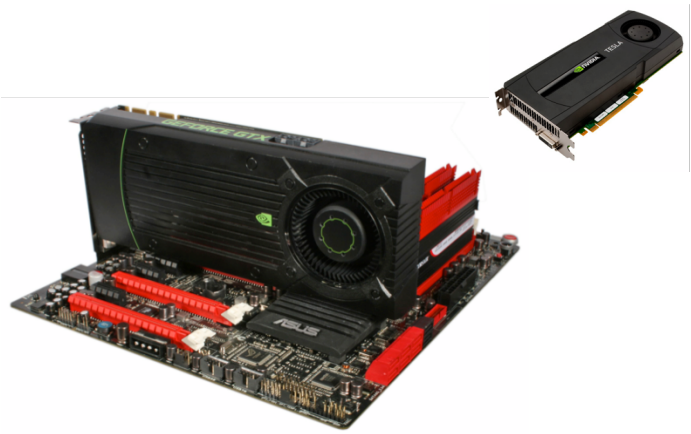
- The Rise of GPU Computing



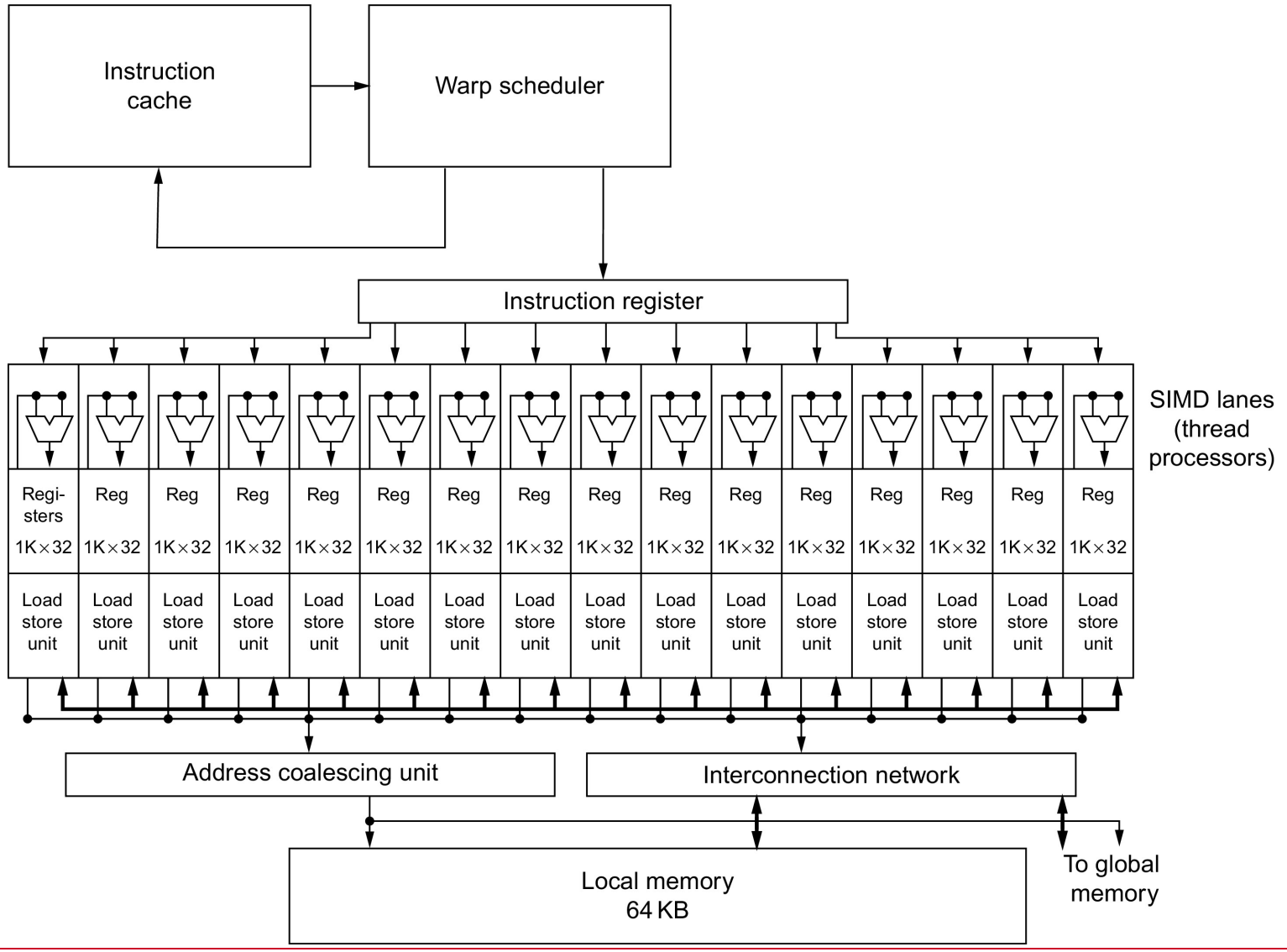
DATA LEVEL PARALLELISM (DLP) - GPU



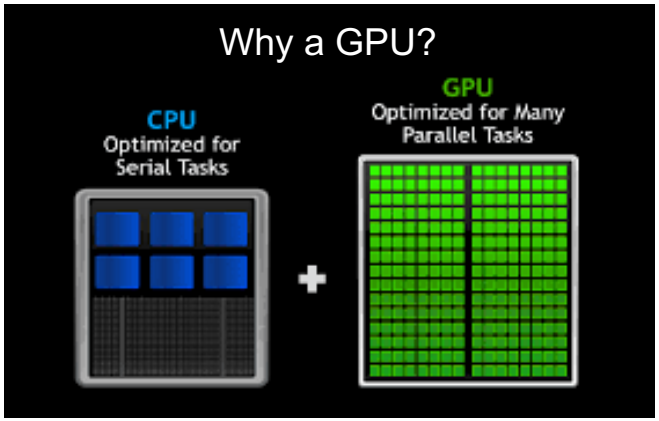


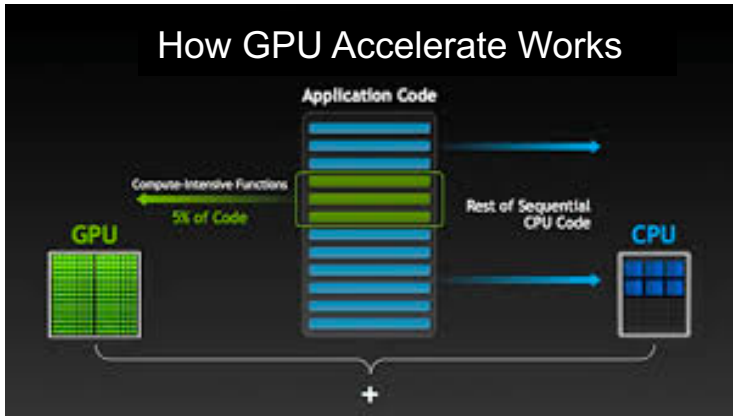


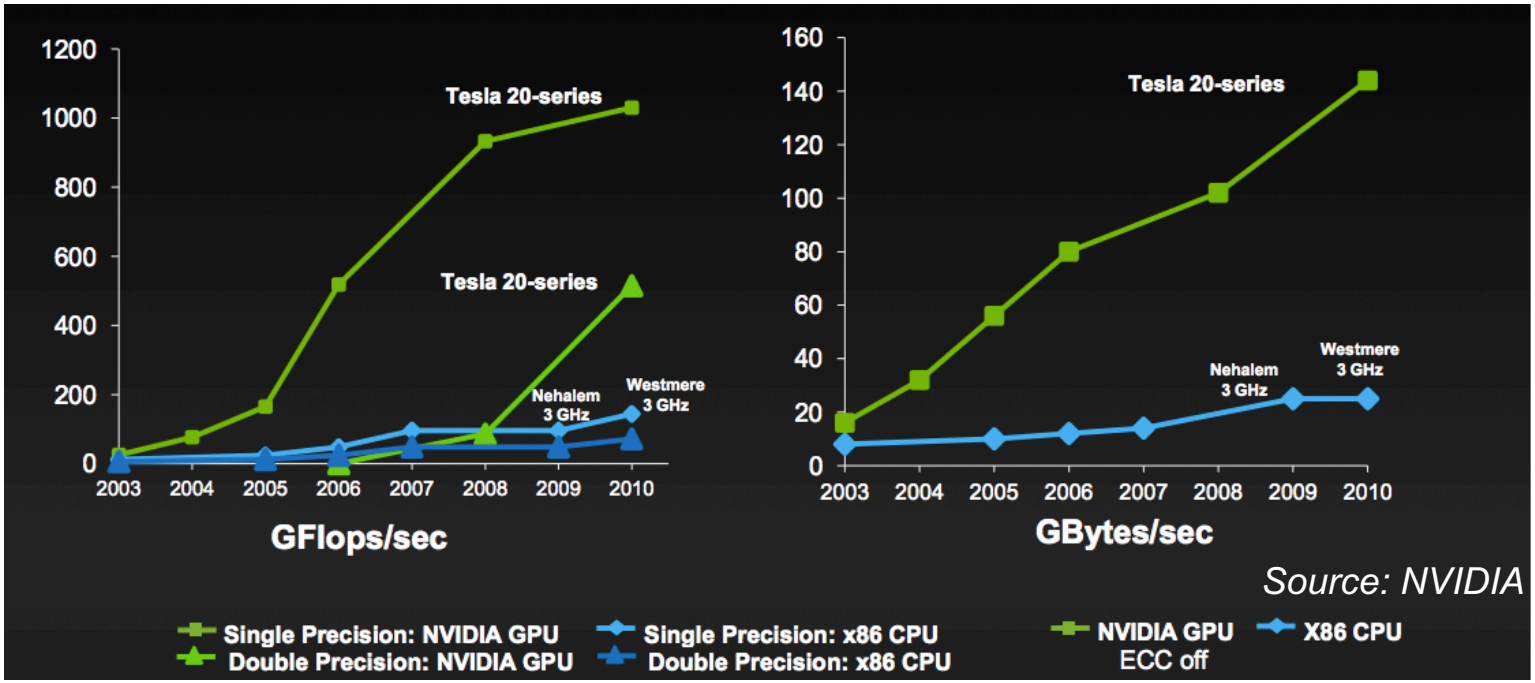
DATA LEVEL PARALLELISM (DLP) - GPU



DATA LEVEL PARALLELISM (DLP) - GPU







DATA LEVEL PARALLELISM (DLP) - GPU

Data Parallelism in GPUs

- GPUs take advantages of massive DLP to provide very high FLOP rates

  -- > 1 Tera DP FLOP in NVIDIA GK110

- "SIMT" execution model

  -- Single instruction multiple threads (SIMT)

  -- Trying to distinguish itself from both "vectors" and "SIMD"

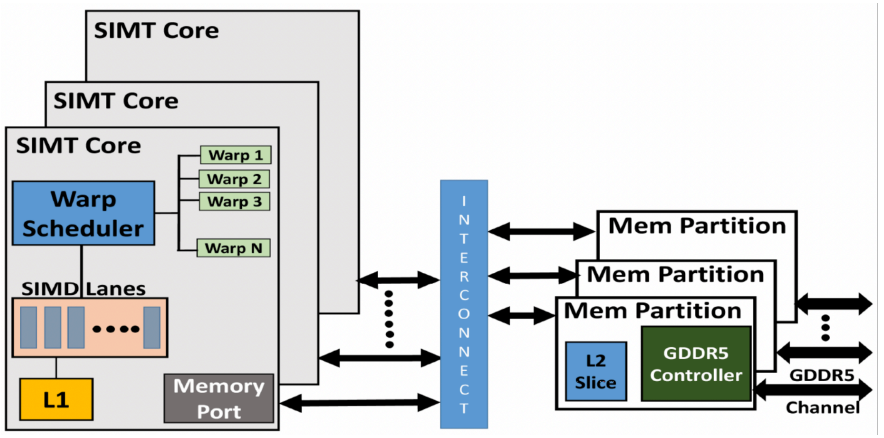
  -- A key difference: better support for conditional control flow

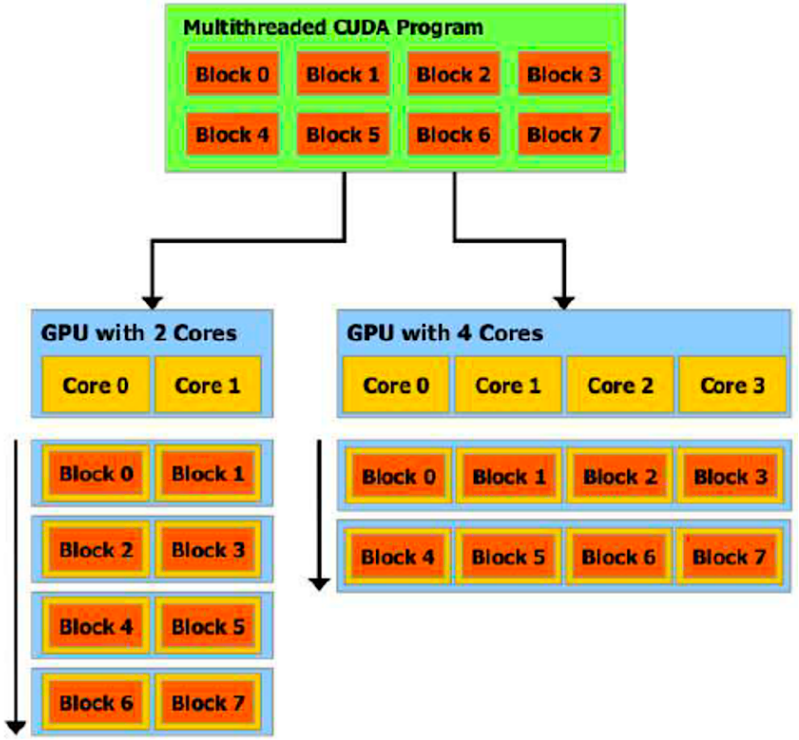
- Program it with CUDA or OpenCL

  -- Extensions to C

  -- Perform a "shader task" (a snippet of scalar computation) over many elements

  -- Internally, GPU uses scatter/gather and vector-mask like operations



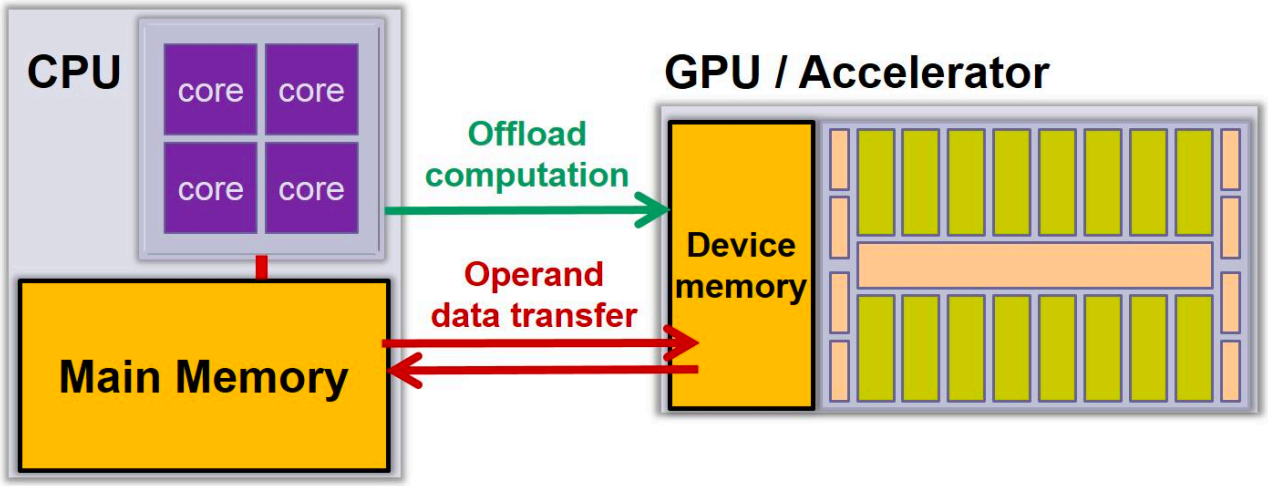


DATA LEVEL PARALLELISM (DLP) - GPU

- It is a processor optimized for 2D/3D graphics, video, visual computing, and display.
- It is highly parallel, highly multithreaded multiprocessor optimized for visual computing.
- It provide real-time visual interaction with computed objects via graphics images, and video.
- It serves as both a programmable graphics processor and a scalable parallel computing platform.
  - Heterogeneous systems: combine a GPU with a CPU
- It is called as Many-core

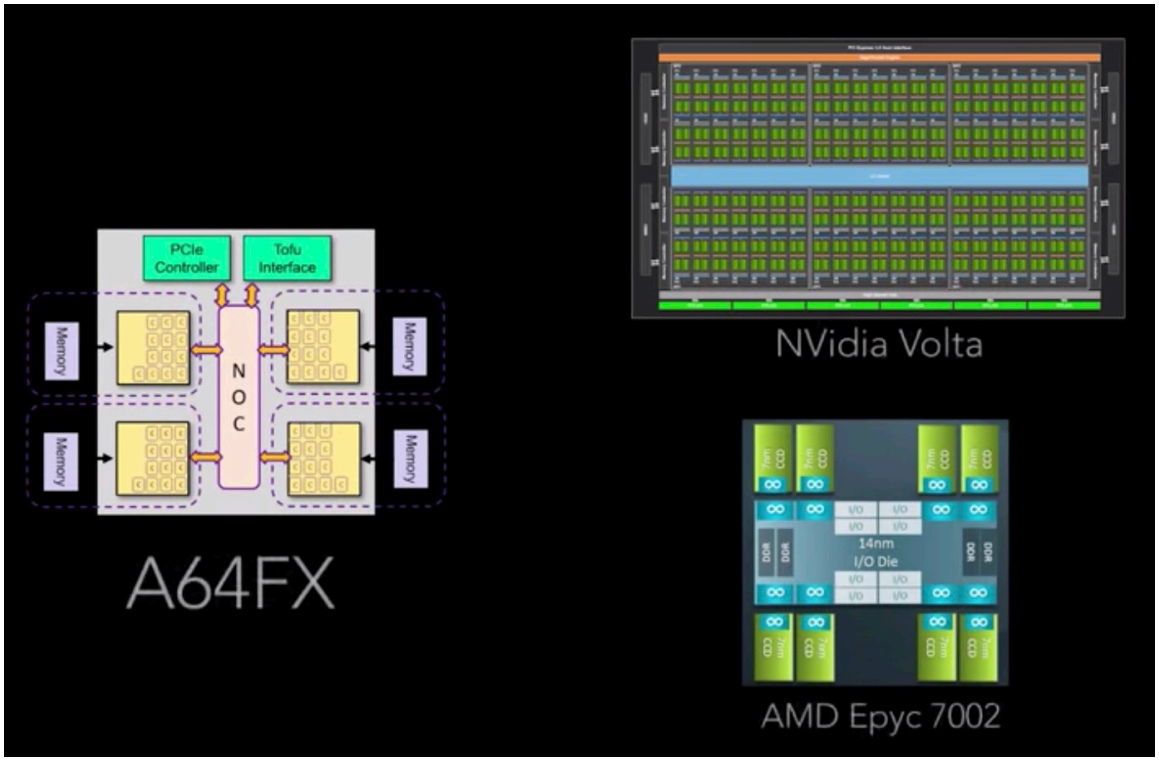
DATA LEVEL PARALLELISM (DLP) - GPU

- A heterogeneous system
  - A regular host CPU
  - A GPU that handles CUDA (may be on the same CPU chip)



DATA LEVEL PARALLELISM (DLP)

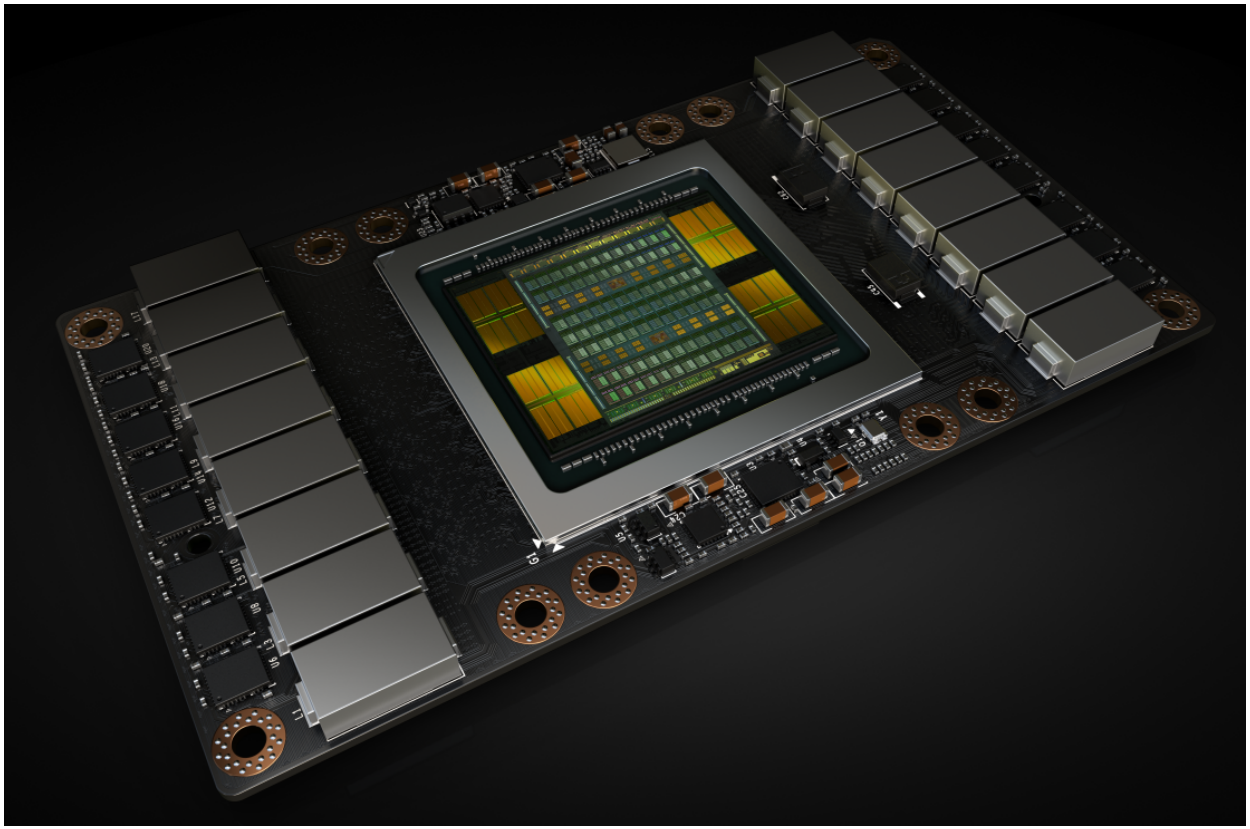
- A heterogeneous system



CPU? GPU? This new ARM chip is BOTH

DATA LEVEL PARALLELISM (DLP) - GPU

- 2017 NVIDIA Volta GV100 GPU



DATA LEVEL PARALLELISM (DLP) - GPU

- 2017 NVIDIA Volta GV100 GPU
  - Released May 2017
    - Total 84 SM
  - Cores
    - 5120 FP32 cores
  - Can do FP16 also
    - 2560 FP64 cores
    - 640 Tensor cores
  - Memory
    - 16G HBM2
    - L2: 6144 KB
    - Shared memory: 96KB * 80 (SM)
    - Register File: 20,480 KB (Huge)

https://devblogs.nvidia.com/parallelforall/inside-volta/

DATA LEVEL PARALLELISM (DLP)

- Vector processor vs. GPUs

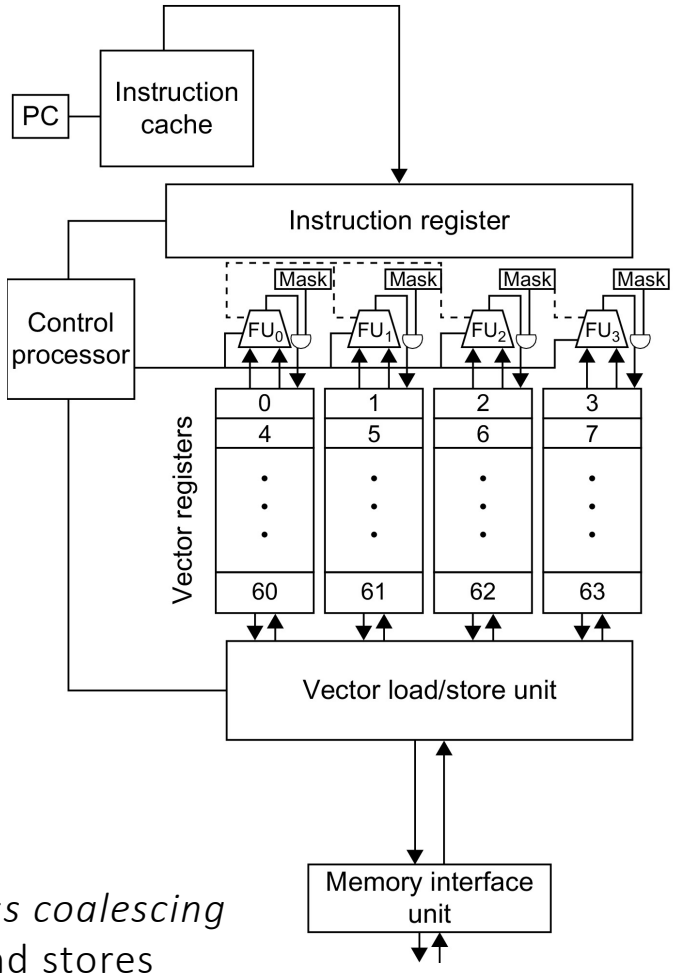
-- A vector processor has a scalar unit

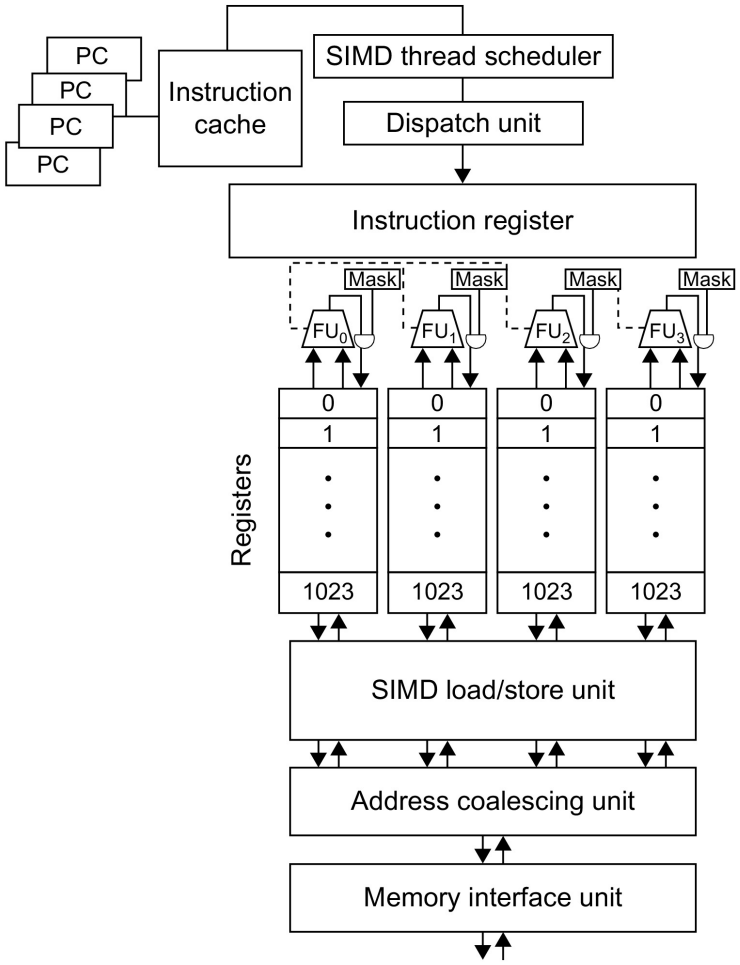
-- A GPU is designed to handle a large number of threads

-- Each SIMD within a GPU has a large number of registers that are dynamically assigned by thread scheduler

-- All load/store operations in a GPU are separate across lanes (No vector)

-- GPUs rely on *address coalescing unit* to merge loads and stores





DATA LEVEL PARALLELISM (DLP)

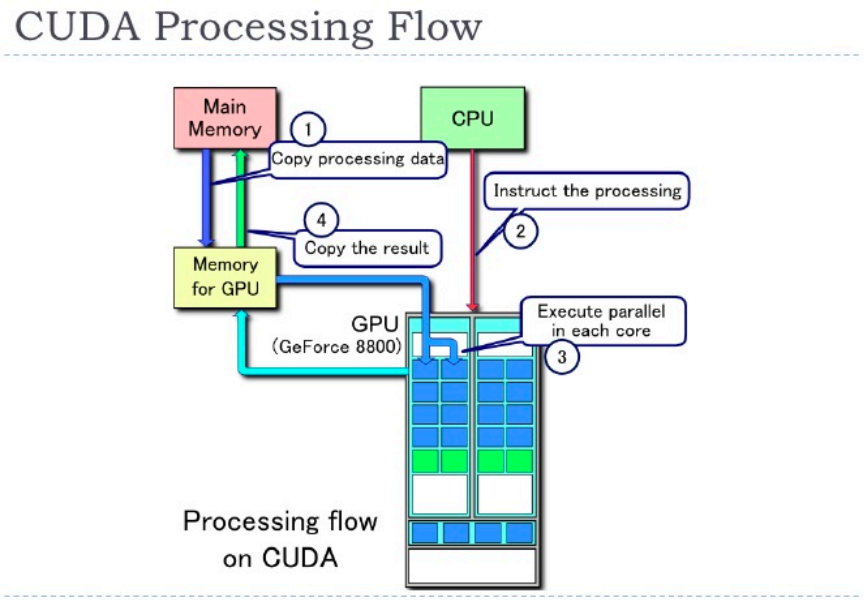
- SIMD ISEs vs. GPUs

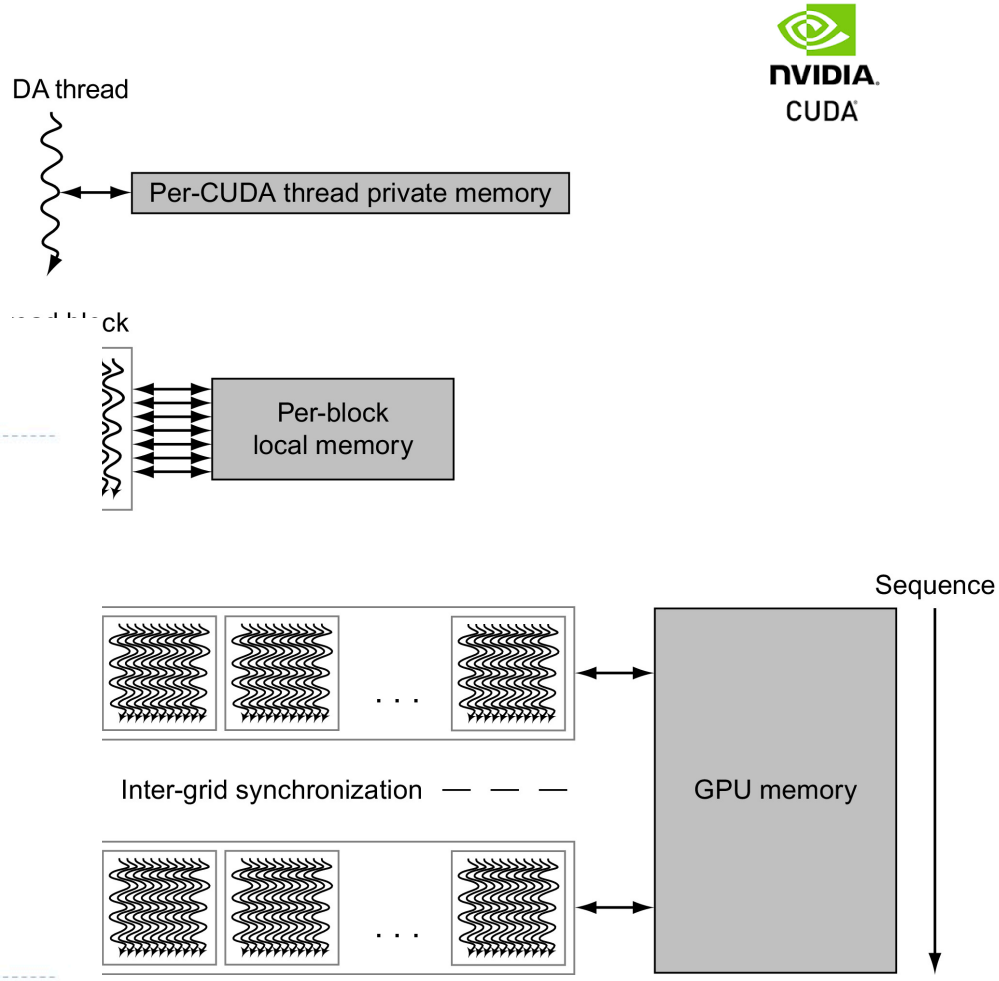
Feature	Multicore with SIMD	GPU
SIMD Processors	4–8	8–32
SIMD Lanes/Processor	2–4	up to 64
Multithreading hardware support for SIMD Threads	2–4	up to 64
Typical ratio of single-precision to double-precision performance	2:1	2:1
Largest cache size	40 MB	4 MB
Size of memory address	64-bit	64-bit
Size of main memory	up to 1024 GB	up to 24 GB
Memory protection at level of page	Yes	Yes
Demand paging	Yes	Yes
Integrated scalar processor/SIMD Processor	Yes	No
Cache coherent	Yes	Yes on some systems

DATA LEVEL PARALLELISM (DLP)

- Programming GPUs
  - NVIDIA has CUDA:

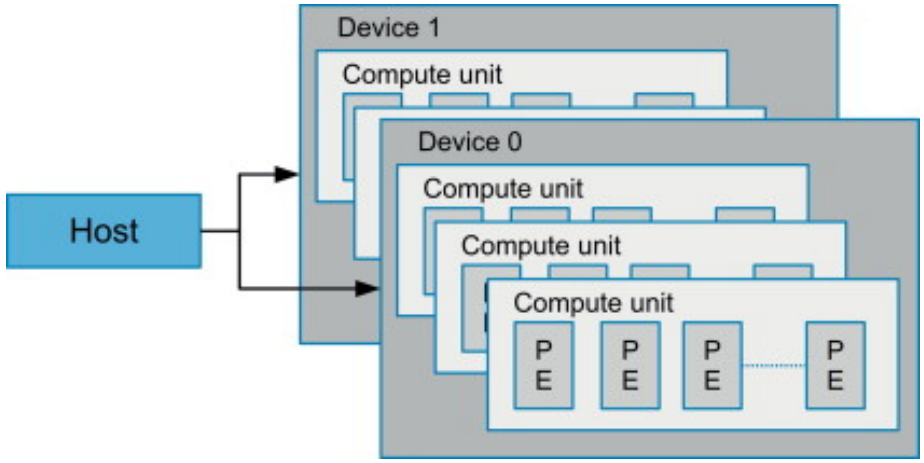
NVIDA GPUs have this abstract instruction set called Parallel Thread Execution (PTX) that allows compatibility amongst various generations of GPUs



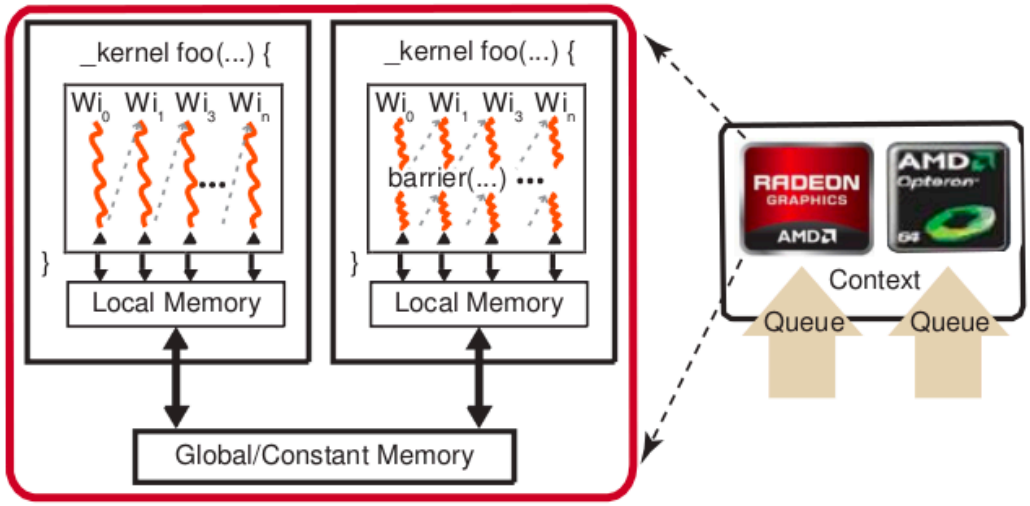


DATA LEVEL PARALLELISM (DLP)

- AMD and others have OpenCL programming



OpenCL Platform Model



OpenCL programming Model

Review

- ILP: Instruction Level Parallelism
  - Out-of-Order execution (all in hardware)
  - Pipeline: overlap processing stages of different instructions
  - SISD, IPC hardly achieves more than 2
- DLP: Data Level Parallelism
  - Vector Processors
  - SIMD extensions
  - GPUs
- TLP: Thread **Level Parallelism**
  - Multiprocessors
  - HW Multithread

Fundamental Problem of ILP

Clock rate, IPC are at odds:

- Pipelining: Fast clock; Increased hazards lower IPC
- Higher IPC; Lower clocks

Fundamental Problem of DLP

- Single operation repeated on multiple data elements
- Less general than ILP: parallel instructions are same operations

Fundamental Problem of TLP

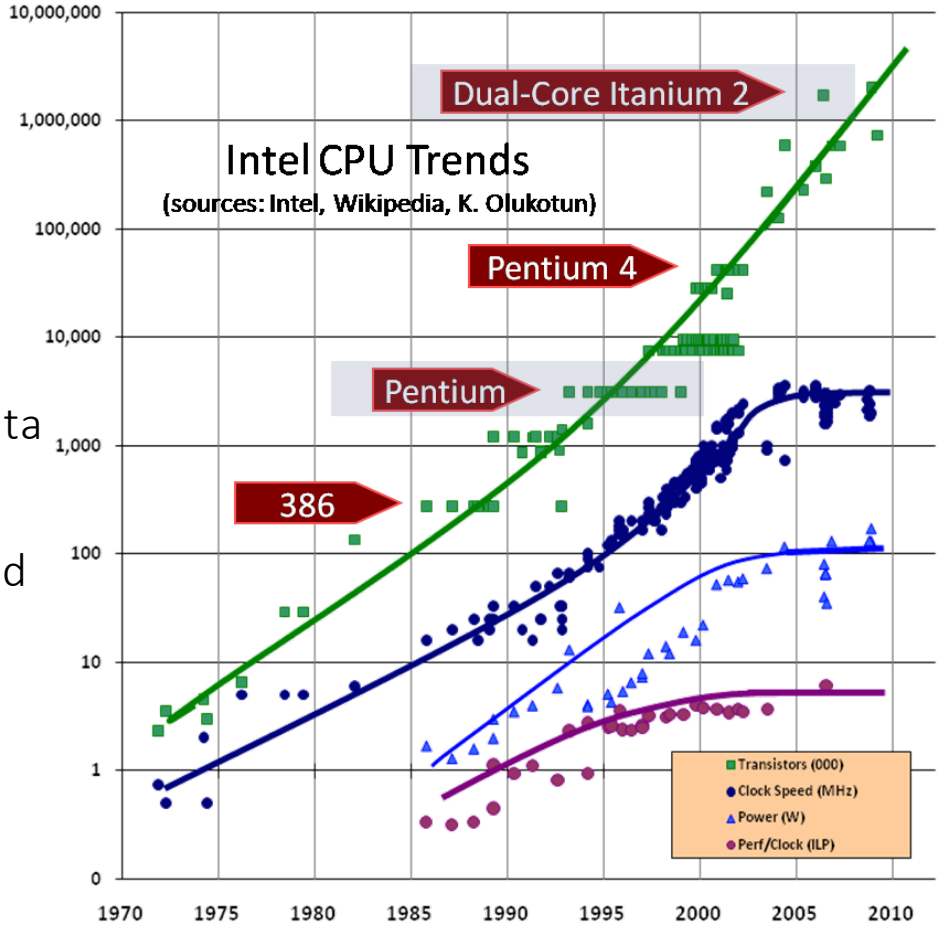
- Power Wall
- Memory Wall
- ILP Wall
- Brick Wall

OVERVIEW

- Motivation for multicore

Need for performance enhancement:

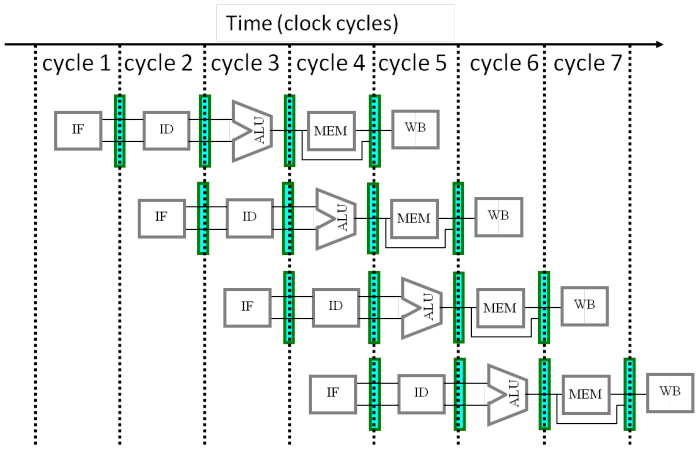
- Need a faster, more capable system to support increasing transmission speed and communication bandwidth
- Increasing resolution of image/video data
- Increasing security need
- Increasing use of video conferencing and surveillance
- New applications, e.g., drug discovery with high volume DNA data processing, scientific computing, weather foresting

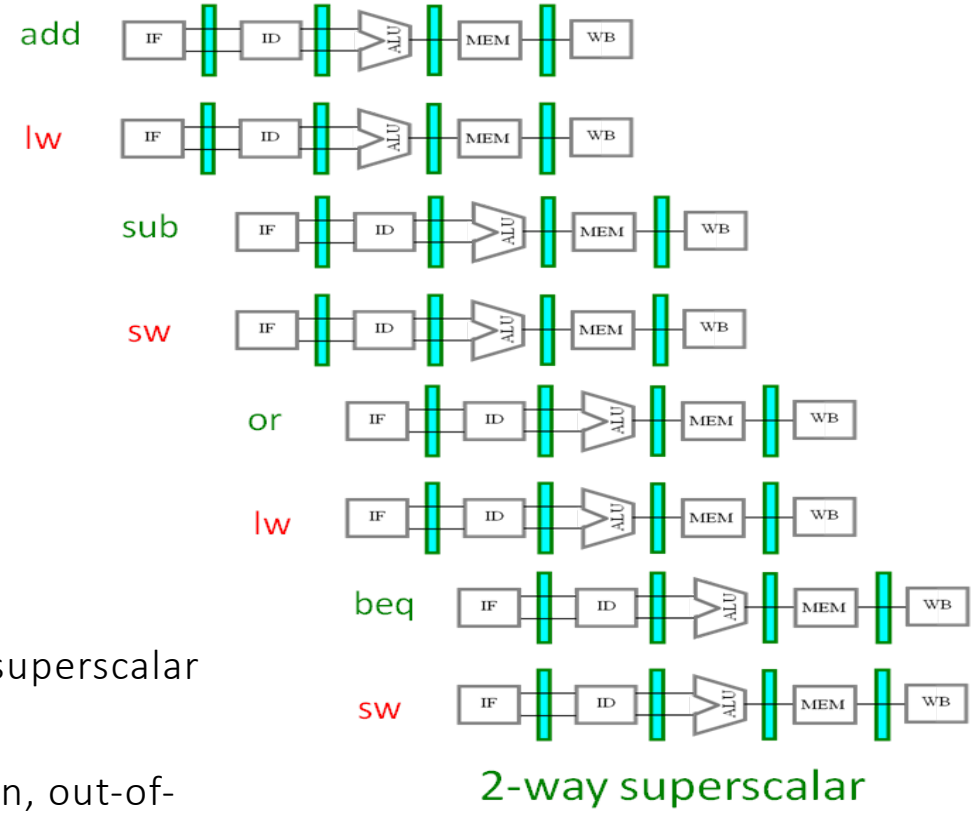


2.5 quintillion (2.5×10^{18}) bytes of data created everyday —sensors and RFID, social network, digital pictures and videos, purchase records, GPS signals, email communications[IBM bigdata]

OVERVIEW

- Traditional approaches for performance enhancement - ILP





1991-2000 was the decade of ILP

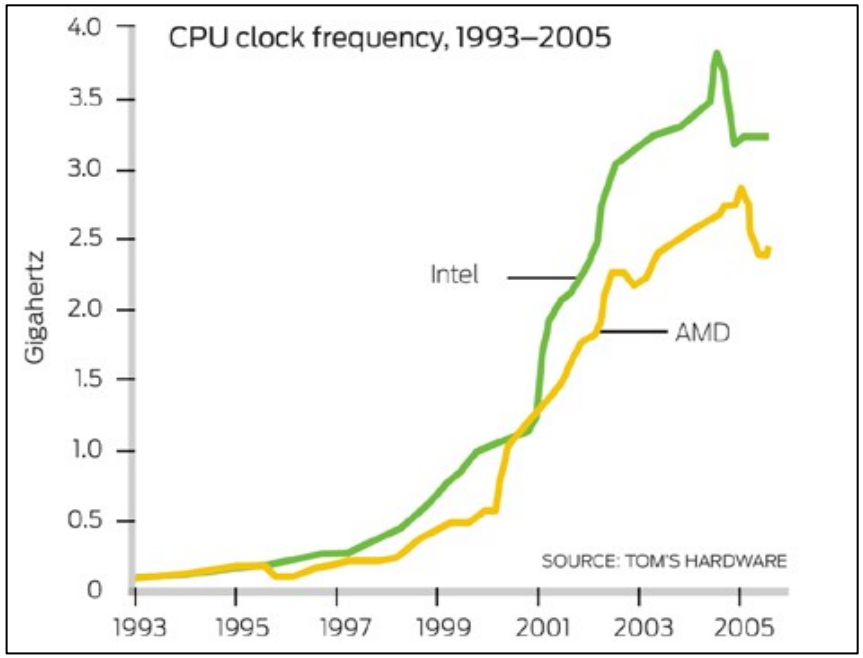
- ✓ Pipelined instruction execution
- ✓ Multiple instructions issued per cycle: superscalar and VLIW processors
- ✓ Branch prediction, speculative execution, out-of-order execution
- ✓ Advanced cache design

OVERVIEW

- Problems
  - The Power Wall

$$P = \alpha \cdot C \cdot V_{dd}^{\,2} \cdot f$$

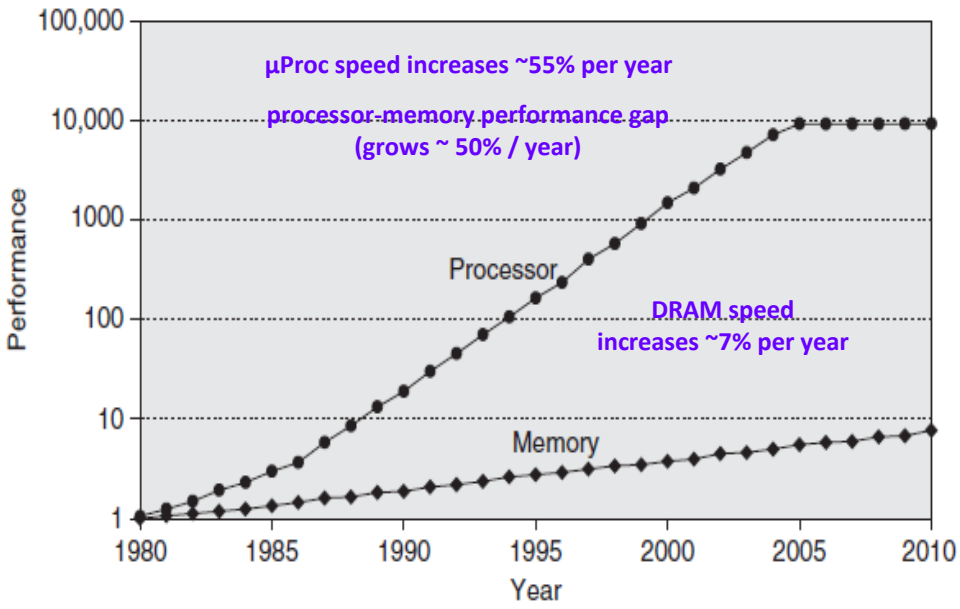
V_{dd} increases with f

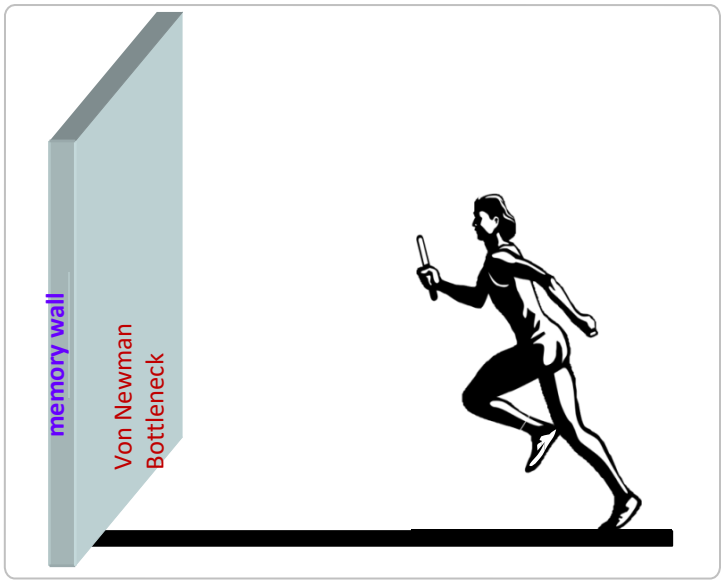


- ✓ Power consumption increases more rapidly with the increasing operating frequency.
- ✓ Around the beginning of 2003, the ever-increasing processor speed was checked due to very high power consumption.
- ✓ Intel Tejas dissipated 150W heat at 2.80GHz: The Tejas had been projected to run 7 GHz. The project was cancelled in May 2004.

OVERVIEW

- Problems
  - The Memory Wall

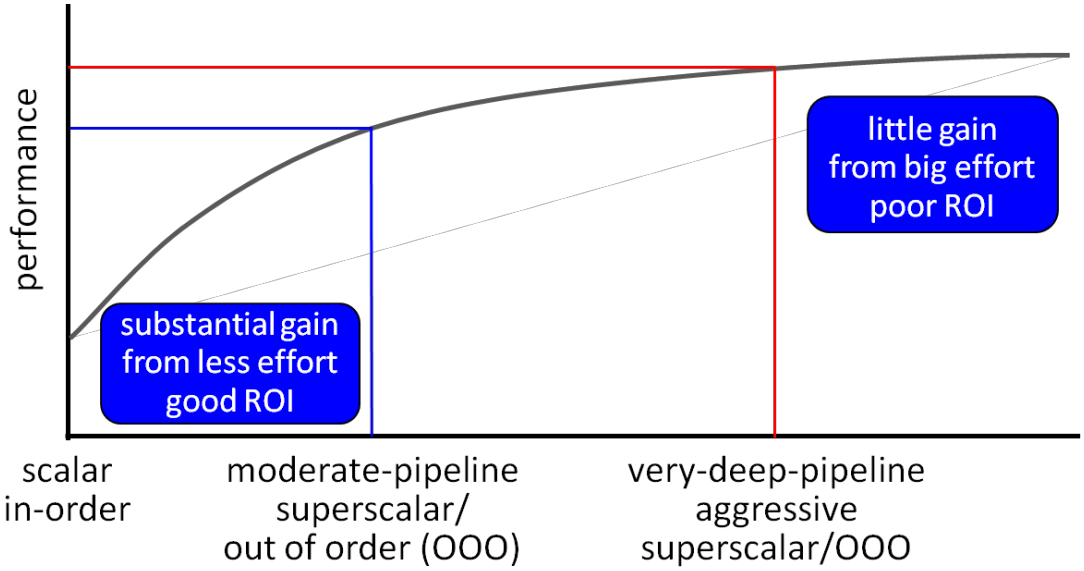




- ✓ Widening gap between compute bandwidth and memory bandwidth could not be bridged - resulting in memory latency.
- ✓ Memory hierarchy was proposed as a solution.
- ✓ But the increase in the size of the on-chip cache and cache optimization no longer yield improvement.

Overview

- Problems
  - The ILP Wall

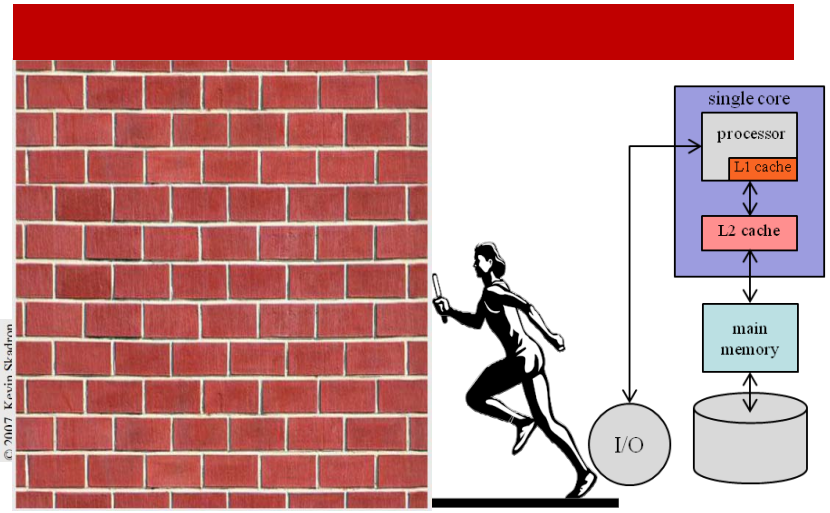


- ✓ High design and verification time and cost
- ✓ Diminishing returns in more ILP
- ✓ Instruction-level parallelism is near its limit

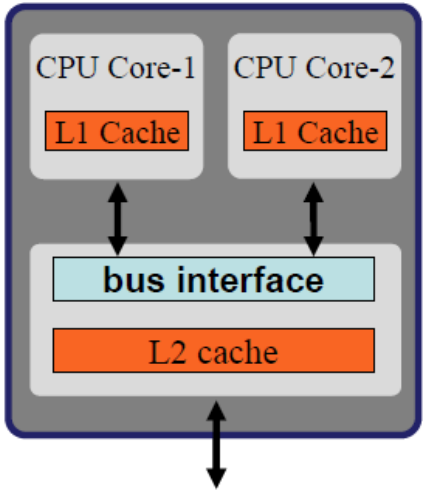
OVERVIEW

- Problems
  - The Brick Wall

Power Wall + Memory Wall + ILP Wall = Brick Wall

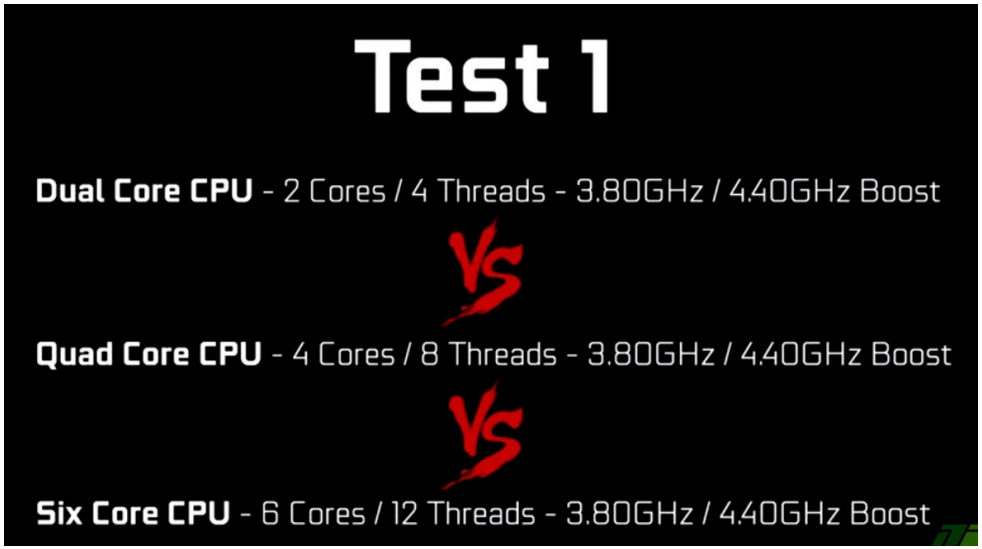


- ✓ It could be possible to overcome the brick wall by a multicore processor.
- ✓ Containing two or more processors in the same chip, that provides enhanced performance by efficient simultaneous processing of multiple tasks and consumes less power due to lower clock frequency.
- ✓ However, more cores…



A generic dual core processor

Thread Level Parallelism (TLP) - Need for Multicore

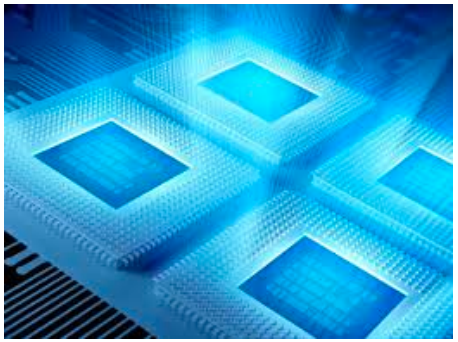


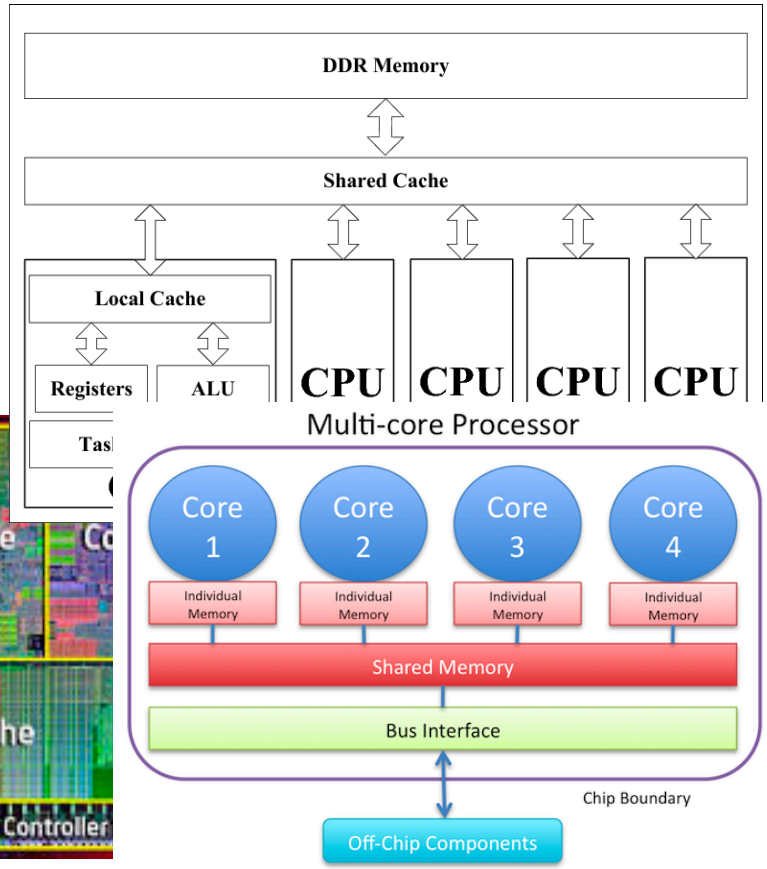
[2 Cores vs. 4 Cores vs. 6 Cores Windows Boot Loading Times Comparison]

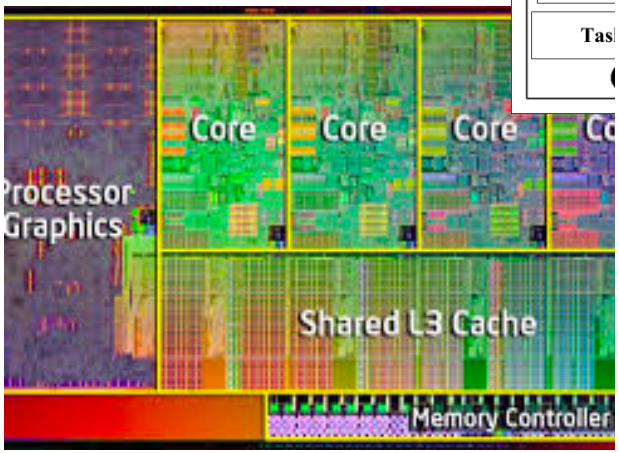
[How Many Cores are Really Needed for Modern Games]

TLP - Need for Multicore

- A multicore processor is a single computing component with two or more "independent" processors (called "cores").







- ✓ Database servers
- ✓ Web servers
- ✓ Telecommunication markets
- ✓ Multimedia applications
- ✓ Scientific applicati on ns
- ✓ In general, applications with Thread-level parallelism (as opposed to instruction-level parallelism)

TLP - Need for Multicore

- Task Parallelism: Several functions on the same data: average, minimum, binary or, geometric mean
- No dependencies between the tasks, so all can run in parallel

- ✓ Overcome **power wall** using multiple slow cores

  -- Cores running at lower clock frequency and lower voltage can still deliver the desired performance using less power

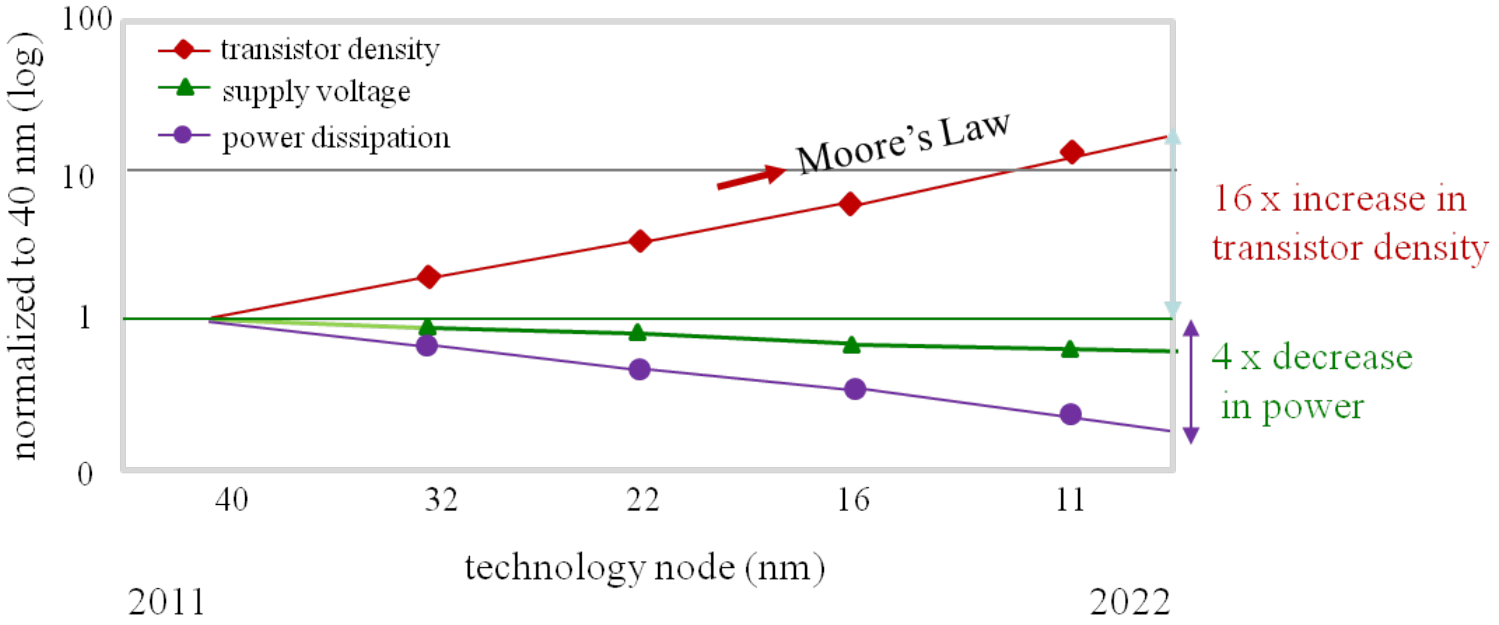
  -- Scale up the number of cores rather than frequency

- ✓ Overcome **memory wall** with memory parallelism
- ✓ **Derive parallelism** by thread-level parallelism
- ✓ **Homogenous multicore systems**: consist of identical cores: less design and verification cost (e.g., ARM quad-core Cortex-A53)
- ✓ **Heterogeneous multicore systems**: contain different types of cores, each optimized for a different role (e.g., Cortex-A, Cortex-M, and DSP cores)

TLP - Need for Multicore

- Challenges

1. A new power wall



Technology scaling continues according to Moore's law

→ increasing transistor density

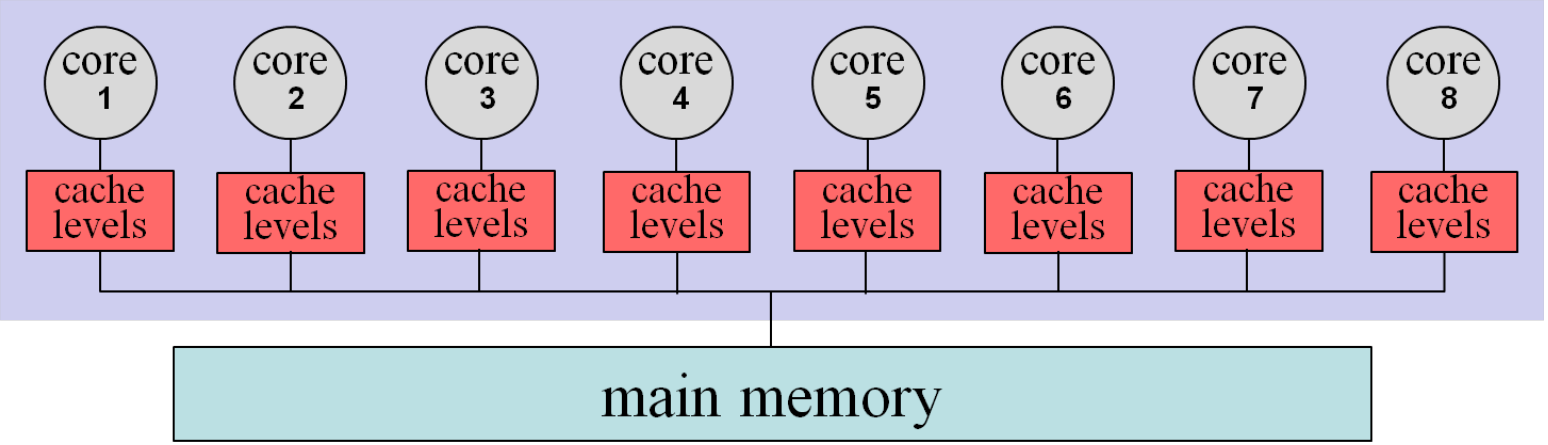
→ more heat dissipation per unit area

The new power wall is going to constrain multicore too.

TLP - Need for Multicore

- Challenges

2. Memory wall



Increase in memory latency is coming as a new memory wall

- As number of cores increases, performance enhancement is degraded:
  - Due to lack of memory bandwidth
  - Contention between cores over the memory bus

TLP - Need for Multicore

- Challenges **3. Interconnection problem**

On-chip interconnects are becoming a critical bottleneck

- Increasing number of cores, interconnect length increases since data moves across the cores on the chip
- Interconnect delay increases with technology: would be reaching above 12 ns for 1 mm of copper wire by 2020
- The interconnect power density has been rapidly increasing with feature size
- Also requires mechanisms for efficient inter-processor coordination
  - Synchronization
  - Mutual exclusion
  - Context switching

Anything more to note?

We Have Learned So Far

- Un-pipelining and Pipelining
  - ✓ Pipeline Stages
  - ✓ Performance comparison
- Instruction Level Parallelism (ILP)
  - ✓ Pipeline Processor
  - ✓ Pipeline Hazards
- Data Level Parallelism (DLP)
- Task Level Parallelism (TLP)

-- *Advanced Topic of Pipelining*

Multicore, the Memory Wall, and Numerical Compression